opsawg Z. Li
Internet-Draft R. Gu
Intended status: Standards Track China Mobile
Expires: November 13, 2018 J. Dong
Huawei Technologies
May 12, 2018

Export BGP community information in IP Flow Information Export (IPFIX)
draft-ietf-opsawg-ipfix-bgp-community-07

Abstract

By introducing new Information Elements (IEs), this draft extends the existing BGP related IEs to enable IPFIX [RFC7011] to export the BGP community information, including the information of BGP standard community [RFC1997], BGP extended community [RFC4360], and BGP large community [RFC8092]. Network traffic information can then be accumulated and analysed at the BGP community granularity, which represents the traffic of different kinds of customers, services, or geographical regions according to the network operator’s BGP community planning. Network traffic information at the BGP community granularity is useful for network traffic analysis and engineering.

To clarify, no new BGP community attribute is defined in this document and this document has no purpose to replace BGP Monitoring Protocol (BMP) defined in RFC7854. The IEs introduced in this document are used by IPFIX together with other IEs to facilitate the IPFIX collector analyzing the network traffic at the BGP community granularity without running the heavy BGP protocol. When needed, the mediator or collector can use the IEs introduced in this document to report the BGP community related traffic flow information it gets either from exporters or through local correlation to other IPFIX devices.

Table of Contents

1. Introduction

IP Flow Information Export (IPFIX) [RFC7011] provides network administrators with traffic flow information using the Information Elements (IEs) defined in [IANA-IPFIX] registries. Based on the traffic flow information, network administrators know the amount and direction of the traffic in their network, then they can optimize their network when needed. For example, they can shift some flows from the congested links to the low utilized links through a SDN controller or PCE [RFC4655].

[IANA-IPFIX] has already defined the following IEs for traffic flow information exporting in different granularities: sourceIPv4Address, sourceIPv4Prefix, destinationIPv4Address, destinationIPv4Prefix, bgpSourceAsNumber, bgpDestinationAsNumber, bgpNextHopIPv4Address, etc. In some circumstances, however, especially when traffic engineering and optimization are executed in the Tier 1 or Tier 2 operators' backbone networks, traffic flow information based on these IEs may not be suitable. Flow information based on IP address or IP prefix may provide much too fine granularity for a large network. On the contrary, flow information based on AS number may be too coarse.

BGP community is a BGP path attribute defined in IDR (Inter Domain Routing) working group. The already defined BGP community attribute includes the standard community defined in [RFC1997], the extended community defined in [RFC4360], and the large community defined in [RFC8092]. BGP community attribute has a variety of use cases, one practice of which is to use BGP community with planned specific values to represent the groups of customers, services, geographical and topological regions, which is used by a lot of operators in their field networks. Please refer to [RFC4384], [RFC8195] and Section 3 of this document for the detailed examples. To know the traffic generated by differnt kinds of customers, from differnt geographical or topological regions, by differnt kinds of customers in differnt regions, we need the corresponding community information related to the traffic flow exported by IPFIX. Netwok traffic statistic at the BGP community granularity is useful not only for the traffic analyzing, but also can then be used by other applications, such as the traffic optimization applications located in IPFIX collector, SDN controller or PCE. [Community-TE] also states analyzing network traffic information at the BGP community granularity is prefered for inbound traffic engineering. However, there is no IE defined for BGP community attribute in [IANA-IPFIX] yet.

Flow information based on BGP community may be collected by a mediator defined in [RFC6183]. Mediator is responsible for the correlation between flow information and BGP community. However no IEs are defined in [RFC6183] for exporting BGP community information in IPFIX. Furthermore, to correlate the BGP community with the flow information, mediator needs to learn BGP routes and perform lookup in the BGP routing table to get the matching entry for a specific flow. Neither BGP route learning nor routing table lookup is trivial for a mediator. Mediator is mainly introduced to release the performance requirement for the exporter [RFC5982]. In fact, to obtain the information for the already defined BGP related IEs, such as bgpSourceAsNumber, bgpDestinationAsNumber, and bgpNextHopIPv4Address, etc, the exporter has to hold the up-to-date BGP routing table and perform lookup in the BGP routing table. The exporter can obtain the BGP community information in the same procedure, thus the additional load added by exporting BGP community information is minimal if the exporter is already exporting the existing BGP related IEs. It is RECOMMENDED that the BGP community information be exported by the exporter directly using IPFIX.

Through running BGP [RFC4271] or BMP [RFC7854] and performing lookup in the BGP routing table to get the matching entry for a specific flow (we call it correlation), IPFIX collectors and other applications, such as SDN controller or PCE, can figure up the network traffic at the BGP community granularity. However, neither running BGP or BMP protocol nor routing table lookup is trivial for the IPFIX collectors and other applications. Moreover correlation between IPFIX flow information and the BGP RIB on the exporter (such as router) is more accurate, compared to the correlation on a collector, since the BGP routing table may be updated when the IPFIX collectors and other applications reveive the IPFIX flow information. And as stated above, the exporter can obtain the BGP community information in the same procedure when it obtains other BGP related informaiton. So exporting the BGP community information directly by the exporter to the collector is the efficient and accurate way. If the IPFIX collectors and other applications only want to figure up the network traffic at the BGP community granularity, they do not need to run the heavy BGP or BMP protocol when the BGP community information can be obtained by IPFIX. However, we have to clarify, the BMP protocol has its own application scenario, the mechanisum introduced in this document has no purpose to replace it.

By introducing new IEs, this draft extends the existing BGP related IEs to enable IPFIX [RFC7011] to export the BGP community information, including BGP standard community defined in [RFC1997], BGP extended community defined in [RFC4360], and BGP large community defined in [RFC8092]. Flow information, including packetDeltaCount, octetDeltaCount [RFC7012] etc, can then be accumulated and analysed by the collector or other applications, such as SDN controller or PCE [RFC4655], at the BGP community granularity, which is useful for knowing the traffic generted by different kinds of customers, from differnt geographical or topological regions according to the operator’s BGP community plan, and can then be used by the traffic engineering or traffic optimization applications, especially in the backbone network.

The IEs introduced in this document are applicable for both IPv4 and IPv6 traffic. Both the exporter and the mediator can use these IEs to export BGP community information in IPFIX. When needed, the mediator or collector can use these IEs to report the BGP community related traffic flow information it gets either from exporters or through local correlation to other IPFIX devices.

To clarify, no new BGP community attribute is defined in this document, IDR (Inter Domain Routing) working group is the right place to define new community attributes for the BGP protocol.

Note that this document does not update the IPFIX specification [RFC7011] and the Information Model [RFC7012] because IANA’s IPFIX registry [IANA-IPFIX] is the ultimate Information Element reference, per Section 1 of [RFC7012].

Please refer [IANA-IPFIX] for the whole list of the already defined BGP related IEs.

Please refer Appendix A for the encoding example and Section 3 for a detailed use case.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. BGP Community based Traffic Collection

[RFC4384] introduces the mechanism of using BGP standard communities and extended communities to collect the geographical and topological related information in BGP routing system. [RFC8195] gives some examples about the application of BGP large communities to represent the geographical regions. Since the network traffic at the BGP community granularity represents the traffic generted by different kinds of customers, from differnt geographical regions according to the network operator’s BGP community plan, it is useful for the network operators to analyze and optimize the network traffic among different customers and regions. This section gives a use case in which the network operator uses the BGP community based traffic information to adjust the network paths for different traffic flows.

Considering the following scenario, AS C provides transit connection between AS A and B. By tagging with different BGP communities, the routes of AS A and B are categorized into several groups respectively with the operator’s plan. For example community A:X and A:Y are used for the routes originated from different geographical regions in AS A, and community B:M and B:N are used for the routes representing the different kinds of customers in AS B, such as B:M is for the mobile customers and B:N is for the fixed line customers. By default, all traffic originated from AS A and destinated to AS B (we call it traffic A-B) goes through path C1-C2-C3 (call it Path-1) in AS C. When the link between C1 and C2 is congested, we cannot simply steer all the traffic A-B from Path-1 to Path C1-C4-C3 (call it Path-2), becuse it will cause the congestion in Path-2.

```
                                +----------+
                                | PCE/SDN  |
                        +-------|Controller|-------+
                        |       +----------+       |
                        |                          |
                        |           AS C           |
                |       |       +----------+       |       |
                |       |   +---|Router C2 |---+   |       |
                |       |   |   +----------+   |   |       |
       AS A     |       |   |100             50|   |       |     AS B
    +--------+  |  +---------+                +---------+  |  +--------+
    |Router A|--|--|Router C1|                |Router C3|--|--|Router B|
    +--------+  |  +---------+                +---------+  |  +--------+
    Community:  |           |100            100|           |  Community:
        A:X     |           |   +----------+   |           |     B:M
        A:Y     |           +---|Router C4 |---+           |     B:N
                                +----------+
```

Figure 1: BGP Community based Traffic Collection

If the PCE/SDN controller in AS C can obtain the network traffic information at the BGP community granularity, it can steer some traffic related to some BGP communities (when we consider only the source or destination of the traffic), or some BGP community pairs (when we consider both the source and the destination of the traffic) from Path-1 to Path-2 according to the utilization of different paths. For instance, steer the traffic generated by community A:X from Path-1 to Path-2 by deploying route policy at Router C1, or steer the traffic from community A:Y to community B:M from Path-1 to Path-2. Using the IEs defined in this document, IPFIX can export the BGP community information related to a specific traffic flow togecher with other flow information. The traffic information can then be accumulated at the BGP community granularity and used by the PCE/SDN controller to steer the appropriate traffic from Path-1 to Path-2.

4. IEs for BGP Standard Community

[RFC1997] defines the BGP Communities attribute, called BGP Standard Community in this document, which describes a group of routes sharing some common properties. BGP Standard Communities are treated as 32 bit values as stated in[RFC1997].

In order to export BGP standard community information along with other flow information defined by IPFIX, three new IEs are introduced. One is bgpCommunity, which is used to identify that the value in this IE is a BGP standard community. The other two are bgpSourceCommunityList and bgpDestinationCommunityList, which are both basicList [RFC6313] of bgpCommunity, and are used to export BGP standard community information corresponding to a specific flow’s source IP and destination IP respectively.

The detailed information of the three new IEs are shown in the following sections.

4.1. bgpCommunity

ElementID	to be assigned by IANA
Name	bgpCommunity
Data Type	unsigned32
Data Type Semantics	identifier
Description	BGP community as defined in [RFC1997]
Units	none

Figure 2: bgpCommunity

4.2. bgpSourceCommunityList

ElementID	to be assigned by IANA
Name	bgpSourceCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP communities corresponding with source IP address of a specific flow
Units	none

Figure 3: bgpSourceCommunityList

4.3. bgpDestinationCommunityList

ElementID	to be assigned by IANA
Name	bgpDestinationCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP communities corresponding with destination IP address of a specific flow
Units	none

Figure 4: bgpDestinationCommunityList

5. IEs for BGP Extended Community

[RFC4360] defines the BGP Extended Communities attribute, which provides a mechanism for labeling the information carried in BGP. Each Extended Community is encoded as an 8-octet quantity with the format defined in [RFC4360].

In order to export BGP Extended Community information together with other flow information by IPFIX, three new IEs are introduced. The first one is bgpExtendedCommunity, which is used to identify that the value in this IE is a BGP Extended Community. The other two are bgpSourceExtendedCommunityList and bgpDestinationExtendedCommunityList, which are both basicList [RFC6313] of bgpExtendedCommunity, and are used to export the BGP Extended Community information corresponding to a specific flow's source IP and destination IP respectively.

The detailed information of the three new IEs are shown in the following sections.

5.1. bgpExtendedCommunity

ElementID	to be assigned by IANA
Name	bgpExtendedCommunity
Data Type	octetArray
Data Type Semantics	default
Description	BGP Extended Community as defined in [RFC4360] The size of this Information Element MUST be 8 octets.
Units	none

Figure 5: bgpExtendedCommunity

5.2. bgpSourceExtendedCommunityList

ElementID	to be assigned by IANA
Name	bgpSourceExtendedCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP Extended Communities corresponding with source IP address of a specific flow
Units	none

Figure 6: bgpSourceExtendedCommunityList

5.3. bgpDestinationExtendedCommunityList

ElementID	to be assigned by IANA
Name	bgpDestinationExtendedCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP Extended communities corresponding with destination IP address of a specific flow
Units	none

Figure 7: bgpDestinationExtendedCommunityList

6. IEs for BGP Large Community

[RFC8092] defines the BGP Large Communities attribute, which is suitable for use with all Autonomous System Numbers (ASNs) including four-octet ASNs. Each BGP Large Community is encoded as a 12-octet quantity with the format defined in [RFC8092].

In order to export BGP Large Community information together with other flow information by IPFIX, three new IEs are introduced. The first one is bgpLargeCommunity, which is used to identify that the value in this IE is a BGP Large Community. The other two are bgpSourceLargeCommunityList and bgpDestinationLargeCommunityList, which are both basicList [RFC6313] of bgpLargeCommunity, and are used to export the BGP Large Community information corresponding to a specific flow's source IP and destination IP respectively.

The detailed information of the three new IEs are shown in the following sections.

6.1. bgpLargeCommunity

ElementID	to be assigned by IANA
Name	bgpLargeCommunity
Data Type	octetArray
Data Type Semantics	default
Description	BGP Large Community as defined in [RFC8092] The size of this Information Element MUST be 12 octets.
Units	none

Figure 8: bgpLargeCommunity

6.2. bgpSourceLargeCommunityList

ElementID	to be assigned by IANA
Name	bgpSourceLargeCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP Large Communities corresponding with source IP address of a specific flow
Units	none

Figure 9: bgpSourceLargeCommunityList

6.3. bgpDestinationLargeCommunityList

ElementID	to be assigned by IANA
Name	bgpDestinationLargeCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP Large communities corresponding with destination IP address of a specific flow
Units	none

Figure 10: bgpDestinationLargeCommunityList

7. Operational Considerations

The maximum length of an IPFIX message is 65535 bytes as per [RFC7011] , and the maximum length of a normal BGP message is 4096 bytes as per [RFC4271]. Since BGP communities, including standard, extended, and large communities , are BGP path attributes carried in BGP Update messages, the total length of these attributes can not exceed the length of a BGP message, i.e. 4096 bytes. So one IPFIX message with maximum length of 65535 bytes has enough space to fit all the communities related to a specific flow, both the source IP and the destination IP related.

[I-D.ietf-idr-bgp-extended-messages] extends the maximum size of a BGP Update message to 65535 bytes. Then theoretically the BGP community information related to a specific flow may exceed the length one IPFIX message. However, according to the information about the networks in the field, the number of BGP communities in one BGP route is usually no more than 10. Nevertheless, BGP speakers that support the extended message SHOULD be careful to export the BGP communities in the IPFIX message properly, such as only convey as many communities as possible in the IPFIX message. The collector which receives an IPFIX message with maximum length and BGP communities contained in its data set SHOULD be aware that the BGP communities may be truncated due to limited message space. In this case, it is RECOMMENDED to configure export policy of BGP communities on the exporter to limit the BGP communities to be exported, so as to only export some specific communities,or not to export some specific communities.

If needed, we may consider to extend the message length of IPFIX [RFC7011] from 16 bits to 32 bits to solve this problem completely. The detailed mechanism is out of the scope of this document.

To align with the size of BGP extended community and large community, the size of IE bgpExtendedCommunity and bgpLargeCommunity is 8 octets and 12 octets respectively. In the event that the bgpExtendedCommunity or bgpLargeCommunity IE is not of its expected size, the IPFIX collector SHOULD ignore it. This is intended to protect implementations using BGP logic from calling their parsing routines with invalid lengths.

For the proper processing of the exporter, when it receives the template requesting to report the BGP community information (refer Appendix A for an example), the exporter SHOULD obtain the corresponding BGP community information through BGP lookup using the corresponding source or destination IP of the specific traffic flow. When exporting the IPFIX information to the collector, the exporter SHOULD include the corresponding BGP communities in the IPFIX message.

8. Security Considerations

This document only defines new IEs for IPFIX. This document itself does not directly introduce security issues. The same security considerations as for the IPFIX Protocol Specification [RFC7011] and Information Model [RFC7012] apply.

As the BGP community information is deducible by other means, there are no increased privacy concerns, neither.

9. IANA Considerations

This draft specifies the following IPFIX IEs to export BGP community information along with other flow information.

The Element IDs for these IEs are solicited to be assigned by IANA. The following table is for IANA's reference to put in each field in the registry.

ElementID	Name	Data Type	Data Type Semantics
TBA1	bgpCommunity	unsigned32	identifier
TBA2	bgpSourceCommunityList	basicList	list
TBA3	bgpDestinationCommunityList	basicList	list
TBA4	bgpExtendedCommunity	octetArray	default
TBA5	bgpSourceExtended CommunityList	basicList	list
TBA6	bgpDestinationExtended CommunityList	basicList	list
TBA7	bgpLargeCommunity	octetArray	default
TBA8	bgpSourceLargeCommunityList	basicList	list
TBA9	bgpDestinationLarge CommunityList	basicList	list

ElementID	Description	Units
TBA1	BGP community as defined in [RFC1997]	
TBA2	zero or more BGP communities corresponding with source IP address of a specific flow	
TBA3	zero or more BGP communities corresponding with destination IP address of a specific flow	

TBA4	BGP Extended Community as defined in [RFC4360] The size of this IE MUST be 8 octets	
TBA5	zero or more BGP Extended Communities corresponding with source IP address of a specific flow	
TBA6	zero or more BGP Extended communities corresponding with destination IP address of a specific flow	
TBA7	BGP Large Community as defined in [RFC8092] The size of this IE MUST be 12 octets.	
TBA8	zero or more BGP Large Communities corresponding with source IP address of a specific flow	
TBA9	zero or more BGP Large communities corresponding with destination IP address of a specific flow	

ElementID	Range	References	Requester	Revision	date
TBA1		RFC1997	this draft	0	
TBA2		RFC6313,RFC1997	this draft	0	
TBA3		RFC6313,RFC1997	this draft	0	
TBA4		RFC4360	this draft	0	
TBA5		RFC6313,RFC4360	this draft	0	
TBA6		RFC6313,RFC4360	this draft	0	
TBA7		RFC8092	this draft	0	
TBA8		RFC6313,RFC8092	this draft	0	
TBA9		RFC6313,RFC8092	this draft	0	

Figure 11: IANA Considerations

10. Acknowledgements

The authors would like to thank Benoit Claise and Paul Aitken for their comments and suggestions to promote this document. Appreciations are given to Tianran Zhou, Warren Kumari, Jeffrey Haas, Ignas Bagdonas, Stewart Bryant, Paolo Lucente, Job Snijders, Jared Mauch, Rudiger Volk, etc, for their discussion, comments and suggestions in the face to face meetings and in the mail list.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, DOI 10.17487/RFC6313, July 2011, <https://www.rfc-editor.org/info/rfc6313>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

11.2. Informative References

[Community-TE] Shao, W., Devienne, F., Iannone, L., and JL. Rougier, "On the use of BGP communities for fine-grained inbound traffic engineering", Computer Science 27392(1):476-487, November 2015.

[I-D.ietf-idr-bgp-extended-messages] Bush, R., Patel, K., and D. Ward, "Extended Message support for BGP", draft-ietf-idr-bgp-extended-messages-24 (work in progress), November 2017.

[IANA-IPFIX] "IP Flow Information Export (IPFIX) Entities", <http://www.iana.org/assignments/ipfix/>.

[RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <https://www.rfc-editor.org/info/rfc1997>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4384] Meyer, D., "BGP Communities for Data Collection", BCP 114, RFC 4384, DOI 10.17487/RFC4384, February 2006, <https://www.rfc-editor.org/info/rfc4384>.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

[RFC5982] Kobayashi, A., Ed. and B. Claise, Ed., "IP Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, DOI 10.17487/RFC5982, August 2010, <https://www.rfc-editor.org/info/rfc5982>.

[RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, DOI 10.17487/RFC6183, April 2011, <https://www.rfc-editor.org/info/rfc6183>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <https://www.rfc-editor.org/info/rfc7012>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

[RFC8092] Heitz, J., Ed., Snijders, J., Ed., Patel, K., Bagdonas, I., and N. Hilliard, "BGP Large Communities Attribute", RFC 8092, DOI 10.17487/RFC8092, February 2017, <https://www.rfc-editor.org/info/rfc8092>.

[RFC8195] Snijders, J., Heasley, J., and M. Schmidt, "Use of BGP Large Communities", RFC 8195, DOI 10.17487/RFC8195, June 2017, <https://www.rfc-editor.org/info/rfc8195>.

Appendix A. Encoding Example

In this section, we give an example to show the encoding format for the new introduced IEs.

Flow information including BGP communities is shown in the below table. Suppose we want all the fields to be reported by IPFIX.

Source IP	Destination IP	BGP community corresponding with Source IP	BGP community corresponding with Destination IP
1.1.1.1	2.2.2.2	1:1001,1:1002,8:1001	2:1002,8:1001
3.3.3.3	4.4.4.4	3:1001,3:1002,8:1001	4:1001,8:1001

Figure 12: Flow information including BGP communities

A.1. Template Record

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          SET ID = 2           |         Length = 24           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Template ID = 256        |        Field Count = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    SourceIPv4Address = 8    |        Field length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| DestinationIPv4Address = 12 |        Field length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| bgpSourceCommunityList= TBA2|     Field length = 0xFFFF     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| bgpDestinationCommunityList |     Field length = 0xFFFF     |
| |         = TBA3              |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Template Record Encoding Format

In this example, the Template ID is 256, which will be used in the Data Record. The field length for bgpSourceCommunityList and bgpDestinationCommunityList is 0xFFFF, which means the length of this

IE is variable, the actual length of this IE is indicated by the list length field in the basic list format as per [RFC6313].

A.2. Data Set

The data set is represented as follows:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          SET ID = 256         |           Length = 92         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                   SourceIPv4Address = 1.1.1.1                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                DestinationIPv4Address = 2.2.2.2               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      255      |         List length = 17      |semantic=allof |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      bgpCommunity = TBA1      |          Field Len = 4        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        BGP Source Community Value 1 = 1:1001                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        BGP Source Community Value 2 = 1:1002                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        BGP Source Community Value 3 = 8:1001                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     255       |         List length = 13      |semantic =allof|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      bgpCommunity = TBA1      |          Field Len = 4        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         BGP Destination Community Value 1 = 2:1002            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         BGP Destination Community Value 2 = 8:1001            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                   SourceIPv4Address = 3.3.3.3                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                DestinationIPv4Address = 4.4.4.4               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     255       |         List length = 17      |semantic =allof|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      bgpCommunity = TBA1      |          Field Len = 4        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        BGP Source Community Value 1  = 3:1001                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        BGP Source Community Value 2  = 3:1002                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        BGP Source Community Value 3  = 8:1001                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   |     255       |        List length = 13       |semantic =allof|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      bgpCommunity = TBA1      |          Field Len = 4        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         BGP Destination Community Value 1 = 4:1001            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         BGP Destination Community Value 2 = 8:1001            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Data Set Encoding Format

Authors' Addresses

Zhenqiang Li
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: li_zhenqiang@hotmail.com

Rong Gu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: gurong_cmcc@outlook.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Network Working Group   E. Lear
Internet-Draft   Cisco Systems
Intended status: Standards Track   R. Droms
Expires: December 17, 2018   Google
                                                           D. Romascanu
                                                           June 15, 2018

Manufacturer Usage Description Specification
draft-ietf-opsawg-mud-25

Abstract

This memo specifies a component-based architecture for manufacturer usage descriptions (MUD). The goal of MUD is to provide a means for end devices to signal to the network what sort of access and network functionality they require to properly function. The initial focus is on access control. Later work can delve into other aspects.

This memo specifies two YANG modules, IPv4 and IPv6 DHCP options, an LLDP TLV, a URL, an X.509 certificate extension and a means to sign and verify the descriptions.

Table of Contents

1. Introduction

The Internet has largely been constructed for general purpose computers, those devices that may be used for a purpose that is specified by those who own the device. [RFC1984] presumed that an end device would be most capable of protecting itself. This made sense when the typical device was a workstation or a mainframe, and it continues to make sense for general purpose computing devices today, including laptops, smart phones, and tablets.

[RFC7452] discusses design patterns for, and poses questions about, smart objects. Let us then posit a group of objects that are specifically not intended to be used for general purpose computing tasks. These devices, which this memo refers to as Things, have a specific purpose. By definition, therefore, all other uses are not intended. If a small number of communication patterns follows from those small number of uses, the combination of these two statements can be restated as a manufacturer usage description (MUD) that can be applied at various points within a network. MUD primarily addresses threats to the device rather than the device as a threat. In some circumstances, however, MUD may offer some protection in the latter case, depending on the MUD-URL is communicated, and how devices and their communications are authenticated.

We use the notion of "manufacturer" loosely in this context to refer to the entity or organization that will state how a device is intended to be used. For example, in the context of a lightbulb, this might indeed be the lightbulb manufacturer. In the context of a smarter device that has a built in Linux stack, it might be an integrator of that device. The key points are that the device itself is assumed to serve a limited purpose, and that there exists an organization in the supply chain of that device that will take responsibility for informing the network about that purpose.

The intent of MUD is to provide the following:

o Substantially reduce the threat surface on a device to those communications intended by the manufacturer.

o Provide a means to scale network policies to the ever-increasing number of types of devices in the network.

o Provide a means to address at least some vulnerabilities in a way that is faster than the time it might take to update systems. This will be particularly true for systems that are no longer supported.

o Keep the cost of implementation of such a system to the bare minimum.

o Provide a means of extensibility for manufacturers to express other device capabilities or requirements.

MUD consists of three architectural building blocks:

o A URL that can be used to locate a description;

o The description itself, including how it is interpreted, and;

o A means for local network management systems to retrieve the description.

MUD is most effective when the network is able to identify in some way the remote endpoints that Things will talk to.

In this specification we describe each of these building blocks and how they are intended to be used together. However, they may also be used separately, independent of this specification, by local deployments for their own purposes.

1.1. What MUD Doesn't Do

MUD is not intended to address network authorization of general purpose computers, as their manufacturers cannot envision a specific communication pattern to describe. In addition, even those devices that have a single or small number of uses might have very broad communication patterns. MUD on its own is not for them either.

Although MUD can provide network administrators with some additional protection when device vulnerabilities exist, it will never replace the need for manufacturers to patch vulnerabilities.

Finally, no matter what the manufacturer specifies in a MUD file, these are not directives, but suggestions. How they are instantiated locally will depend on many factors and will be ultimately up to the local network administrator, who must decide what is appropriate in a given circumstances.

1.2. A Simple Example

A light bulb is intended to light a room. It may be remotely controlled through the network, and it may make use of a rendezvous service of some form that an application on a smart phone. What we can say about that light bulb, then, is that all other network access is unwanted. It will not contact a news service, nor speak to the refrigerator, and it has no need of a printer or other devices. It has no social networking friends. Therefore, an access list applied to it that states that it will only connect to the single rendezvous service will not impede the light bulb in performing its function, while at the same time allowing the network to provide both it and other devices an additional layer of protection.

1.3. Terminology

MUD: manufacturer usage description.

MUD file: a file containing YANG-based JSON that describes a Thing and associated suggested specific network behavior.

MUD file server: a web server that hosts a MUD file.

MUD manager: the system that requests and receives the MUD file from the MUD server. After it has processed a MUD file, it may direct changes to relevant network elements.

MUD controller: a synonym that has been used in the past for MUD manager.

MUD URL: a URL that can be used by the MUD manager to receive the MUD file.

Thing: the device emitting a MUD URL.

Manufacturer: the entity that configures the Thing to emit the MUD URL and the one who asserts a recommendation in a MUD file. The manufacturer might not always be the entity that constructs a Thing. It could, for instance, be a systems integrator, or even a component provider.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.4. Determining Intended Use

The notion of intended use is in itself not new. Network administrators apply access lists every day to allow for only such use. This notion of white listing was well described by Chapman and Zwicky in [FW95]. Profiling systems that make use of heuristics to identify types of systems have existed for years as well.

A Thing could just as easily tell the network what sort of access it requires without going into what sort of system it is. This would, in effect, be the converse of [RFC7488]. In seeking a general solution, however, we assume that a device will implement functionality necessary to fulfill its limited purpose. This is basic economic constraint. Unless the network would refuse access to such a device, its developers would have no reason to provide the network any information. To date, such an assertion has held true.

1.5. Finding A Policy: The MUD URL

Our work begins with the device emitting a Universal Resource Locator (URL) [RFC3986]. This URL serves both to classify the device type and to provide a means to locate a policy file.

MUD URLs MUST use the HTTPS scheme [RFC7230].

In this memo three means are defined to emit the MUD URL, as follows:

- A DHCP option[RFC2131],[RFC3315] that the DHCP client uses to inform the DHCP server. The DHCP server may take further actions, such as act as the MUD manager or otherwise pass the MUD URL along to the MUD manager.

- An X.509 constraint. The IEEE has developed [IEEE8021AR] that provides a certificate-based approach to communicate device characteristics, which itself relies on [RFC5280]. The MUD URL extension is non-critical, as required by IEEE 802.1AR. Various means may be used to communicate that certificate, including Tunnel Extensible Authentication Protocol (TEAP) [RFC7170].

- Finally, a Link Layer Discovery Protocol (LLDP) frame is defined [IEEE8021AB].

It is possible that there may be other means for a MUD URL to be learned by a network. For instance, some devices may already be fielded or have very limited ability to communicate a MUD URL, and yet can be identified through some means, such as a serial number or a public key. In these cases, manufacturers may be able to map those identifiers to particular MUD URLs (or even the files themselves). Similarly, there may be alternative resolution mechanisms available for situations where Internet connectivity is limited or does not exist. Such mechanisms are not described in this memo, but are possible. Implementors are encouraged to allow for this sort of flexibility of how MUD URLs may be learned.

1.6. Processing of the MUD URL

MUD managers that are able to do so SHOULD retrieve MUD URLs and signature files as per [RFC7230], using the GET method [RFC7231]. They MUST validate the certificate using the rules in [RFC2818], Section 3.1.

Requests for MUD URLs SHOULD include an "Accept" header ([RFC7231], Section 5.3.2) containing "application/mud+json", an "Accept-Language" header field ([RFC7231], Section 5.3.5), and a "User-Agent" header ([RFC7231], Section 5.5.3).

MUD managers SHOULD automatically process 3xx response status codes.

If a MUD manager is not able to fetch a MUD URL, other means MAY be used to import MUD files and associated signature files. So long as the signature of the file can be validated, the file can be used. In such environments, controllers SHOULD warn administrators when cache-validity expiry is approaching so that they may check for new files.

It may not be possible for a MUD manager to retrieve a MUD file at any given time. Should a MUD manager fail to retrieve a MUD file, it SHOULD consider the existing one safe to use, at least for a time. After some period, it SHOULD log that it has been unable to retrieve the file. There may be very good reasons for such failures, including the possibility that the MUD manager is in an off-line environment, the local Internet connection has failed, or the remote Internet connection has failed. It is also possible that an attacker is attempting to interfere with the deployment of a device. It is a local decision as to how to handle such circumstances.

1.7. Types of Policies

When the MUD URL is resolved, the MUD manager retrieves a file that describes what sort of communications a device is designed to have. The manufacturer may specify either specific hosts for cloud based services or certain classes for access within an operational network. An example of a class might be "devices of a specified manufacturer type", where the manufacturer type itself is indicated simply by the authority component (e.g, the domain name) of the MUD URL. Another example might be to allow or disallow local access. Just like other policies, these may be combined. For example:

- o Allow access to devices of the same manufacturer
- o Allow access to and from controllers via Constrained Application Protocol (COAP)[RFC7252]
- o Allow access to local DNS/NTP
- o Deny all other access

A printer might have a description that states:

- o Allow access for port IPP or port LPD
- o Allow local access for port HTTP
- o Deny all other access

In this way anyone can print to the printer, but local access would be required for the management interface.

The files that are retrieved are intended to be closely aligned to existing network architectures so that they are easy to deploy. We make use of YANG [RFC7950] because it provides accurate and adequate models for use by network devices. JSON[RFC8259] is used as a serialization format for compactness and readability, relative to XML. Other formats may be chosen with later versions of MUD.

While the policy examples given here focus on access control, this is not intended to be the sole focus. By structuring the model described in this document with clear extension points, other descriptions could be included. One that often comes to mind is quality of service.

The YANG modules specified here are extensions of [I-D.ietf-netmod-acl-model]. The extensions to this model allow for a manufacturer to express classes of systems that a manufacturer would find necessary for the proper function of the device. Two modules are specified. The first module specifies a means for domain names to be used in ACLs so that devices that have their controllers in the cloud may be appropriately authorized with domain names, where the mapping of those names to addresses may rapidly change.

The other module abstracts away IP addresses into certain classes that are instantiated into actual IP addresses through local processing. Through these classes, manufacturers can specify how the device is designed to communicate, so that network elements can be configured by local systems that have local topological knowledge. That is, the deployment populates the classes that the manufacturer specifies. The abstractions below map to zero or more hosts, as follows:

Manufacturer: A device made by a particular manufacturer, as identified by the authority component of its MUD URL

same-manufacturer: Devices that have the same authority component of their MUD URL.

controller: Devices that the local network administrator admits to the particular class.

my-controller: Devices intended to serve as controllers for the MUD-URL that the Thing emitted.

local: The class of IP addresses that are scoped within some administrative boundary. By default it is suggested that this be the local subnet.

The "manufacturer" classes can be easily specified by the manufacturer, whereas controller classes are initially envisioned to be specified by the administrator.

Because manufacturers do not know who will be using their devices, it is important for functionality referenced in usage descriptions to be relatively ubiquitous and mature. For these reasons the YANG-based configuration in a MUD file is limited to either the modules specified or referenced in this document, or those specified in documented extensions.

1.8. The Manufacturer Usage Description Architecture

With these components laid out we now have the basis for an architecture. This leads us to ASCII art.

```
 .......................................
 .                      ____________    .           _____________
 .                     |            |   .          |             |
 .                     |    MUD     |-->get URL-->|    MUD      |
 .                     |  Manager   |   .(https)   | File Server |
 .  End system network |____________|<-MUD file<-<|_____________|
 .                             .        .
 .                             .        .
 . _______                 _________    .
 .|       |  (dhcp et al) | router  |   .
 .| Thing |---->MUD URL-->|   or    |   .
 .|_______|               | switch  |   .
 .                        |_________|   .
 .......................................
```

Figure 1: MUD Architecture

In the above diagram, the switch or router collects MUD URLs and forwards them to the MUD manager (a network management system) for processing. This happens in different ways, depending on how the URL is communicated. For instance, in the case of DHCP, the DHCP server might receive the URL and then process it. In the case of IEEE 802.1X [IEEE8021X], the switch would carry the URL via a certificate to the authentication server via EAP over Radius[RFC3748], which would then process it. One method to do this is TEAP, described in [RFC7170]. The certificate extension is described below.

The information returned by the MUD file server is valid for as long as the Thing is connected. There is no expiry. However, if the MUD manager has detected that the MUD file for a Thing has changed, it SHOULD update the policy expeditiously, taking into account whatever approval flow is required in a deployment. In this way, new recommendations from the manufacturer can be processed in a timely fashion.

The information returned by the MUD file server (a web server) is valid for the duration of the Thing's connection, or as specified in the description. Thus if the Thing is disconnected, any associated configuration in the switch can be removed. Similarly, from time to time the description may be refreshed, based on new capabilities or communication patterns or vulnerabilities.

The web server is typically run by or on behalf of the manufacturer. Its domain name is that of the authority found in the MUD URL. For legacy cases where Things cannot emit a URL, if the switch is able to determine the appropriate URL, it may proxy it. In the trivial case it may hardcode MUD-URL on a switch port or a map from some available identifier such as an L2 address or certificate hash to a MUD-URL.

The role of the MUD manager in this environment is to do the following:

- receive MUD URLs,
- fetch MUD files,
- translate abstractions in the MUD files to specific network element configuration,
- maintain and update any required mappings of the abstractions, and
- update network elements with appropriate configuration.

A MUD manager may be a component of a AAA or network management system. Communication within those systems and from those systems to network elements is beyond the scope of this memo.

1.9. Order of operations

As mentioned above, MUD contains architectural building blocks, and so order of operation may vary. However, here is one clear intended example:

1. Thing emits URL.

2. That URL is forwarded to a MUD manager by the nearest switch (how this happens depends on the way in which the MUD URL is emitted).

3. The MUD manager retrieves the MUD file and signature from the MUD file server, assuming it doesn't already have copies. After validating the signature, it may test the URL against a web or domain reputation service, and it may test any hosts within the file against those reputation services, as it deems fit.

4. The MUD manager may query the administrator for permission to add the Thing and associated policy. If the Thing is known or the Thing type is known, it may skip this step.

5. The MUD manager instantiates local configuration based on the abstractions defined in this document.

6. The MUD manager configures the switch nearest the Thing. Other systems may be configured as well.

7. When the Thing disconnects, policy is removed.

2. The MUD Model and Semantic Meaning

A MUD file consists of a YANG model instance that has been serialized in JSON [RFC7951]. For purposes of MUD, the nodes that can be modified are access lists as augmented by this model. The MUD file is limited to the serialization of only the following YANG schema:

- ietf-access-control-list [I-D.ietf-netmod-acl-model]

- ietf-mud (this document)

- ietf-acldns (this document)

Extensions may be used to add additional schema. This is described further on.

To provide the widest possible deployment, publishers of MUD files SHOULD make use of the abstractions in this memo and avoid the use of IP addresses. A MUD manager SHOULD NOT automatically implement any MUD file that contains IP addresses, especially those that might have local significance. The addressing of one side of an access list is implicit, based on whether it is applied as to-device-policy or from-device-policy.

With the exceptions of "name" of the ACL, "type", "name" of the ACE, and TCP and UDP source and destination port information, publishers of MUD files SHOULD limit the use of ACL model leaf nodes expressed to those found in this specification. Absent any extensions, MUD files are assumed to implement only the following ACL model features:

- match-on-ipv4, match-on-ipv6, match-on-tcp, match-on-udp, match-on-icmp

Furthermore, only "accept" or "drop" actions SHOULD be included. A MUD manager MAY choose to interpret "reject" as "drop". A MUD manager SHOULD ignore all other actions. This is because manufacturers do not have sufficient context within a local deployment to know whether reject is appropriate. That is a decision that should be left to a network administrator.

Given that MUD does not deal with interfaces, the support of the "ietf-interfaces" module [RFC8343] is not required. Specifically, the support of interface-related features and branches (e.g., interface-attachment and interface-stats) of the ACL YANG module is not required.

In fact, MUD managers MAY ignore any particular component of a description or MAY ignore the description in its entirety, and SHOULD carefully inspect all MUD descriptions. Publishers of MUD files MUST NOT include other nodes except as described in Section 3.9. See that section for more information.

2.1. The IETF-MUD YANG Module

This module is structured into three parts:

o The first component, the "mud" container, holds information that is relevant to retrieval and validity of the MUD file itself, as well as policy intended to and from the Thing.

o The second component augments the matching container of the ACL model to add several nodes that are relevant to the MUD URL, or otherwise abstracted for use within a local environment.

o The third component augments the tcp-acl container of the ACL model to add the ability to match on the direction of initiation of a TCP connection.

A valid MUD file will contain two root objects, a "mud" container and an "acls" container. Extensions may add additional root objects as required. As a reminder, when parsing acls, elements within a "match" block are logically ANDed. In general, a single abstraction in a match statement should be used. For instance, it makes little sense to match both "my-controller" and "controller" with an argument, since they are highly unlikely to be the same value.

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is explained in [RFC8340].

```
module: ietf-mud
  +--rw mud!
     +--rw mud-version           uint8
     +--rw mud-url               inet:uri
     +--rw last-update           yang:date-and-time
     +--rw mud-signature?        inet:uri
     +--rw cache-validity?       uint8
     +--rw is-supported          boolean
     +--rw systeminfo?           string
     +--rw mfg-name?             string
     +--rw model-name?           string
     +--rw firmware-rev?         string
     +--rw software-rev?         string
     +--rw documentation?        inet:uri
     +--rw extensions*           string
     +--rw from-device-policy
     |  +--rw acls
     |     +--rw access-list* [name]
     |        +--rw name    -> /acl:acls/acl/name
     +--rw to-device-policy
        +--rw acls
           +--rw access-list* [name]
              +--rw name    -> /acl:acls/acl/name

  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw mud
       +--rw manufacturer?        inet:host
       +--rw same-manufacturer?   empty
       +--rw model?               inet:uri
       +--rw local-networks?      empty
       +--rw controller?          inet:uri
       +--rw my-controller?       empty
  augment
    /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches
       /acl:l4/acl:tcp/acl:tcp:
    +--rw direction-initiated?   direction
```

3. MUD model definitions for the root mud container

3.1. mud-version

This node specifies the integer version of the MUD specification. This memo specifies version 1.

3.2. mud-url

This URL identifies the MUD file. This is useful when the file and associated signature are manually uploaded, say, in an offline mode.

3.3. to-device-policy and from-device-policy containers

[I-D.ietf-netmod-acl-model] describes access-lists. In the case of MUD, a MUD file must be explicit in describing the communication pattern of a Thing, and that includes indicating what is to be permitted or denied in either direction of communication. Hence each of these containers indicates the appropriate direction of a flow in association with a particular Thing. They contain references to specific access-lists.

3.4. last-update

This is a date-and-time value of when the MUD file was generated. This is akin to a version number. Its form is taken from [RFC6991] which, for those keeping score, in turn was taken from Section 5.6 of [RFC3339], which was taken from [ISO.8601.1988].

3.5. cache-validity

This uint8 is the period of time in hours that a network management station MUST wait since its last retrieval before checking for an update. It is RECOMMENDED that this value be no less than 24 and MUST NOT be more than 168 for any Thing that is supported. This period SHOULD be no shorter than any period determined through HTTP caching directives (e.g., "cache-control" or "Expires"). N.B., expiring of this timer does not require the MUD manager to discard the MUD file, nor terminate access to a Thing. See Section 16 for more information.

3.6. is-supported

This boolean is an indication from the manufacturer to the network administrator as to whether or not the Thing is supported. In this context a Thing is said to not be supported if the manufacturer intends never to issue a firmware or software update to the Thing or never update the MUD file. A MUD manager MAY still periodically check for updates.

3.7. systeminfo

This is a textual UTF-8 description of the Thing to be connected. The intent is for administrators to be able to see a brief displayable description of the Thing. It SHOULD NOT exceed 60 characters worth of display space.

3.8. mfg-name, software-rev, model-name firmware-rev

These optional fields are filled in as specified by [RFC8348]. Note that firmware-rev and software-rev MUST NOT be populated in a MUD file if the device can be upgraded but the MUD-URL cannot be. This would be the case, for instance, with MUD-URLs that are contained in 802.1AR certificates.

3.9. extensions

This optional leaf-list names MUD extensions that are used in the MUD file. Note that MUD extensions MUST NOT be used in a MUD file without the extensions being declared. Implementations MUST ignore any node in this file that they do not understand.

Note that extensions can either extend the MUD file as described in the previous paragraph, or they might reference other work. An extension example can be found in Appendix C.

4. Augmentation to the ACL Model

Note that in this section, when we use the term "match" we are referring to the ACL model "matches" node.

4.1. manufacturer

This node consists of a hostname that would be matched against the authority component of another Thing's MUD URL. In its simplest form "manufacturer" and "same-manufacturer" may be implemented as access-lists. In more complex forms, additional network capabilities may be used. For example, if one saw the line "manufacturer" : "flobbidy.example.com", then all Things that registered with a MUD URL that contained flobbity.example.com in its authority section would match.

4.2. same-manufacturer

This null-valued node is an equivalent for when the manufacturer element is used to indicate the authority that is found in another Thing's MUD URL matches that of the authority found in this Thing's MUD URL. For example, if the Thing's MUD URL were https://b1.example.com/ThingV1, then all devices that had MUD URL with an authority section of b1.example.com would match.

4.3. documentation

This URI consists of a URL that points to documentation relating to the device and the MUD file. This can prove particularly useful when the "controller" class is used, so that its use can be explained.

4.4. model

This string matches the entire MUD URL, thus covering the model that is unique within the context of the authority. It may contain not only model information, but versioning information as well, and any other information that the manufacturer wishes to add. The intended use is for devices of this precise class to match, to permit or deny communication between one another.

4.5. local-networks

This null-valued node expands to include local networks. Its default expansion is that packets must not traverse toward a default route that is received from the router. However, administrators may expand the expression as is appropriate in their deployments.

4.6. controller

This URI specifies a value that a controller will register with the MUD manager. The node then is expanded to the set of hosts that are so registered. This node may also be a URN. In this case, the URN describes a well known service, such as DNS or NTP, that has been standardized. Both of those URNs may be found in Section 17.6.

When "my-controller" is used, it is possible that the administrator will be prompted to populate that class for each and every model. Use of "controller" with a named class allows the user to populate that class only once for many different models that a manufacturer may produce.

Controller URIs MAY take the form of a URL (e.g. "http[s]://"). However, MUD managers MUST NOT resolve and retrieve such files, and it is RECOMMENDED that there be no such file at this time, as their form and function may be defined at a point in the future. For now, URLs should serve simply as class names and may be populated by the local deployment administrator.

Great care should be taken by MUD managers when invoking the controller class in the form of URLs. For one thing, it requires some understanding by the administrator as to when it is appropriate. Pre-registration in such classes by controllers with the MUD server is encouraged. The mechanism to do that is beyond the scope of this work.

4.7. my-controller

This null-valued node signals to the MUD manager to use whatever mapping it has for this MUD URL to a particular group of hosts. This may require prompting the administrator for class members. Future work should seek to automate membership management.

4.8. direction-initiated

This MUST only be applied to TCP. This matches the direction in which a TCP connection is initiated. When direction initiated is "from-device", packets that are transmitted in the direction of a thing MUST be dropped unless the thing has first initiated a TCP connection. By way of example, this node may be implemented in its simplest form by looking at naked SYN bits, but may also be implemented through more stateful mechanisms.

When applied this matches packets when the flow was initiated in the corresponding direction. [RFC6092] specifies IPv6 guidance best practices. While that document is scoped specifically to IPv6, its contents are applicable for IPv4 as well.

5. Processing of the MUD file

To keep things relatively simple in addition to whatever definitions exist, we also apply two additional default behaviors:

- o Anything not explicitly permitted is denied.

- o Local DNS and NTP are, by default, permitted to and from the Thing.

An explicit description of the defaults can be found in Appendix B. These are applied AFTER all other explicit rules. Thus, a default behavior can be changed with a "drop" action.

6. What does a MUD URL look like?

MUD URLs are required to use the HTTPS scheme, in order to establish the MUD file server’s identity and assure integrity of the MUD file.

Any "https://" URL can be a MUD URL. For example:

```
     https://things.example.org/product_abc123/v5
     https://www.example.net/mudfiles/temperature_sensor/
     https://example.com/lightbulbs/colour/v1
```

A manufacturer may construct a MUD URL in any way, so long as it makes use of the "https" schema.

7. The MUD YANG Model

```
   <CODE BEGINS>file "ietf-mud@2018-06-15.yang"
   module ietf-mud {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-mud";
     prefix ietf-mud;

     import ietf-access-control-list {
       prefix acl;
     }
     import ietf-yang-types {
       prefix yang;
     }
     import ietf-inet-types {
       prefix inet;
     }

     organization
       "IETF OPSAWG (Ops Area) Working Group";
     contact
       "WG Web: http://tools.ietf.org/wg/opsawg/
        WG List: opsawg@ietf.org
        Author: Eliot Lear
        lear@cisco.com
        Author: Ralph Droms
        rdroms@gmail.com
        Author: Dan Romascanu
        dromasca@gmail.com

       ";
     description
       "This YANG module defines a component that augments the
        IETF description of an access list.  This specific module
        focuses on additional filters that include local, model,
        and same-manufacturer.

        This module is intended to be serialized via JSON and stored
        as a file, as described in RFC XXXX [RFC Editor to fill in with
        this document #].
```

```
        Copyright (c) 2016,2017 IETF Trust and the persons
        identified as the document authors.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD
        License set forth in Section 4.c of the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2018-06-15 {
       description
         "Initial proposed standard.";
       reference
         "RFC XXXX: Manufacturer Usage Description
          Specification";
     }

     typedef direction {
       type enumeration {
         enum to-device {
           description
             "packets or flows destined to the target
              Thing";
         }
         enum from-device {
           description
             "packets or flows destined from
              the target Thing";
         }
       }
       description
         "Which way are we talking about?";
     }

     container mud {
       presence "Enabled for this particular MUD URL";
       description
         "MUD related information, as specified
          by RFC-XXXX [RFC Editor to fill in].";
       uses mud-grouping;
     }

     grouping mud-grouping {
       description
         "Information about when support end(ed), and
          when to refresh";
```

```
       leaf mud-version {
         type uint8;
         mandatory true;
         description
           "This is the version of the MUD
            specification.  This memo specifies version 1.";
       }
       leaf mud-url {
         type inet:uri;
         mandatory true;
         description
           "This is the MUD URL associated with the entry found
            in a MUD file.";
       }
       leaf last-update {
         type yang:date-and-time;
         mandatory true;
         description
           "This is intended to be when the current MUD file
            was generated.  MUD Managers SHOULD NOT check
            for updates between this time plus cache validity";
       }
       leaf mud-signature {
         type inet:uri;
         description
           "A URI that resolves to a signature as
            described in this specification.";
       }
       leaf cache-validity {
         type uint8 {
           range "1..168";
         }
         units "hours";
         default "48";
         description
           "The information retrieved from the MUD server is
            valid for these many hours, after which it should
            be refreshed.  N.B. MUD manager implementations
            need not discard MUD files beyond this period.";
       }
       leaf is-supported {
         type boolean;
         mandatory true;
         description
           "This boolean indicates whether or not the Thing is
            currently supported by the manufacturer.";
       }
       leaf systeminfo {
```

```
         type string;
         description
           "A UTF-8 description of this Thing.  This
            should be a brief description that may be
            displayed to the user to determine whether
            to allow the Thing on the
            network.";
       }
       leaf mfg-name {
         type string;
         description
           "Manufacturer name, as described in
            the ietf-hardware YANG module.";
       }
       leaf model-name {
         type string;
         description
           "Model name, as described in the
            ietf-hardware YANG module.";
       }
       leaf firmware-rev {
         type string;
         description
           "firmware-rev, as described in the
            ietf-hardware YANG module.  Note this field MUST
            NOT be included when the device can be updated
            but the MUD-URL cannot.";
       }
       leaf software-rev {
         type string;
         description
           "software-rev, as described in the
            ietf-hardware YANG module.  Note this field MUST
            NOT be included when the device can be updated
            but the MUD-URL cannot.";
       }
       leaf documentation {
         type inet:uri;
         description
           "This URL points to documentation that
            relates to this device and any classes that it uses
            in its MUD file.  A caution: MUD managers need
            not resolve this URL on their own, but rather simply
            provide it to the administrator.  Parsing HTML is
            not an intended function of a MUD manager.";
       }
       leaf-list extensions {
         type string {
```

```
           length "1..40";
         }
         description
           "A list of extension names that are used in this MUD
            file.  Each name is registered with the IANA and
            described in an RFC.";
       }
       container from-device-policy {
         description
           "The policies that should be enforced on traffic
            coming from the device. These policies are not
            necessarily intended to be enforced at a single
            point, but may be rendered by the controller to any
            relevant enforcement points in the network or
            elsewhere.";
         uses access-lists;
       }
       container to-device-policy {
         description
           "The policies that should be enforced on traffic
            going to the device. These policies are not
            necessarily intended to be enforced at a single
            point, but may be rendered by the controller to any
            relevant enforcement points in the network or
            elsewhere.";
         uses access-lists;
       }
     }

     grouping access-lists {
       description
         "A grouping for access lists in the context of device
          policy.";
       container access-lists {
         description
           "The access lists that should be applied to traffic
              to or from the device.";
         list access-list {
           key "name";
           description
             "Each entry on this list refers to an ACL that
                should be present in the overall access list
                data model. Each ACL is identified by name and
                type.";
           leaf name {
             type leafref {
               path "/acl:acls/acl:acl/acl:name";
             }
```

```
             description
               "The name of the ACL for this entry.";
           }
         }
       }
     }

     augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
       description
         "adding abstractions to avoid need of IP addresses";
       container mud {
         description
           "MUD-specific matches.";
         leaf manufacturer {
           type inet:host;
           description
             "A domain that is intended to match the authority
              section of the MUD URL. This node is used to specify
              one or more manufacturers a device should
              be authorized to access.";
         }
         leaf same-manufacturer {
           type empty;
           description
             "This node matches the authority section of the MUD URL
              of a Thing.  It is intended to grant access to all
              devices with the same authority section.";
         }
         leaf model {
           type inet:uri;
           description
             "Devices of the specified model type will match if
              they have an identical MUD URL.";
         }
         leaf local-networks {
           type empty;
           description
             "IP addresses will match this node if they are
              considered local addresses.  A local address may be
              a list of locally defined prefixes and masks
              that indicate a particular administrative scope.";
         }
         leaf controller {
           type inet:uri;
           description
             "This node names a class that has associated with it
              zero or more IP addresses to match against.  These
              may be scoped to a manufacturer or via a standard
```

```
              URN.";
         }
         leaf my-controller {
           type empty;
           description
             "This node matches one or more network elements that
              have been configured to be the controller for this
              Thing, based on its MUD URL.";
         }
       }
     }
     augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
       "/acl:l4/acl:tcp/acl:tcp" {
       description
         "add direction-initiated";
       leaf direction-initiated {
         type direction;
         description
           "This node matches based on which direction a
            connection was initiated. The means by which that
            is determined is discussed in this document.";
       }
     }
   }
   <CODE ENDS>
```

8. The Domain Name Extension to the ACL Model

This module specifies an extension to IETF-ACL model such that domain names may be referenced by augmenting the "matches" node. Different implementations may deploy differing methods to maintain the mapping between IP address and domain name, if indeed any are needed. However, the intent is that resources that are referred to using a name should be authorized (or not) within an access list.

The structure of the change is as follows:

```
   module: ietf-acldns
     augment /acl:acls/acl:acl/acl:aces/acl:ace/
       acl:matches/acl:l3/acl:ipv4/acl:ipv4:
       +--rw src-dnsname?   inet:host
       +--rw dst-dnsname?   inet:host
     augment /acl:acls/acl:acl/acl:aces/acl:ace/
       acl:matches/acl:l3/acl:ipv6/acl:ipv6:
       +--rw src-dnsname?   inet:host
       +--rw dst-dnsname?   inet:host
```

The choice of these particular points in the access-list model is based on the assumption that we are in some way referring to IP-related resources, as that is what the DNS returns. A domain name in our context is defined in [RFC6991]. The augmentations are replicated across IPv4 and IPv6 to allow MUD file authors the ability to control the IP version that the Thing may utilize.

The following node are defined.

8.1. src-dnsname

The argument corresponds to a domain name of a source as specified by inet:host. A number of means may be used to resolve hosts. What is important is that such resolutions be consistent with ACLs required by Things to properly operate.

8.2. dst-dnsname

The argument corresponds to a domain name of a destination as specified by inet:host See the previous section relating to resolution.

Note when using either of these with a MUD file, because access is associated with a particular Thing, MUD files MUST NOT contain either a src-dnsname in an ACL associated with from-device-policy or a dst-dnsname associated with to-device-policy.

8.3. The ietf-acldns Model

```
<CODE BEGINS>file "ietf-acldns@2018-06-15.yang"
module ietf-acldns {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acldns";
  prefix ietf-acldns;

  import ietf-access-control-list {
    prefix acl;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact
    "WG Web: http://tools.ietf.org/wg/opsawg/
     WG List: opsawg@ietf.org
     Author: Eliot Lear
```

```
          lear@cisco.com
          Author: Ralph Droms
          rdroms@gmail.com
          Author: Dan Romascanu
          dromasca@gmail.com
         ";
       description
         "This YANG module defines a component that augments the
          IETF description of an access list to allow DNS names
          as matching criteria.";

       revision 2018-06-15 {
         description
           "Base version of dnsname extension of ACL model";
         reference
           "RFC XXXX: Manufacturer Usage Description
            Specification";
       }

       grouping dns-matches {
         description
           "Domain names for matching.";
         leaf src-dnsname {
           type inet:host;
           description
             "domain name to be matched against";
         }
         leaf dst-dnsname {
           type inet:host;
           description
             "domain name to be matched against";
         }
       }

       augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
        "/acl:l3/acl:ipv4/acl:ipv4" {
         description
           "Adding domain names to matching";
         uses dns-matches;
       }
       augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
        "/acl:l3/acl:ipv6/acl:ipv6" {
         description
           "Adding domain names to matching";
         uses dns-matches;
       }
     }
     <CODE ENDS>
```

9. MUD File Example

This example contains two access lists that are intended to provide outbound access to a cloud service on TCP port 443.


```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://lighting.example.com/lightbulb2000",
    "last-update": "2018-03-02T11:20:51+01:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "The BMS Example Lightbulb",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6fr"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6to"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-76100-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:src-dnsname": "test.example.com",
                  "protocol": 6
                },
                "tcp": {
                  "ietf-mud:direction-initiated": "from-device",
```

```
                  "source-port": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-76100-v6fr",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-frdev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:dst-dnsname": "test.example.com",
                  "protocol": 6
                },
                "tcp": {
                  "ietf-mud:direction-initiated": "from-device",
                  "destination-port": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```


In this example, two policies are declared, one from the Thing and the other to the Thing. Each policy names an access list that applies to the Thing, and one that applies from. Within each access list, access is permitted to packets flowing to or from the Thing that can be mapped to the domain name of "service.bms.example.com". For each access list, the enforcement point should expect that the Thing initiated the connection.

10. The MUD URL DHCP Option

The IPv4 MUD URL client option has the following format:

```
+------+-----+------------------------------
| code | len |  MUDstring
+------+-----+------------------------------
```

Code OPTION_MUD_URL_V4 (161) is assigned by IANA. len is a single octet that indicates the length of MUD string in octets. The MUD string is defined as follows:

```
MUDstring = mudurl [ " " reserved ]
mudurl = URI; a URL [RFC3986] that uses the "https" schema [RFC7230]
reserved = 1*( OCTET ) ; from [RFC5234]
```

The entire option MUST NOT exceed 255 octets. If a space follows the MUD URL, a reserved string that will be defined in future specifications follows. MUD managers that do not understand this field MUST ignore it.

The IPv6 MUD URL client option has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_MUD_URL_V6     |        option-length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            MUDstring                          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

OPTION_MUD_URL_V6 (112; assigned by IANA).

option-length contains the length of the MUDstring, as defined above, in octets.

The intent of this option is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview of the network system as managed by the network administrator to decide what to do with this information. The key function of this option is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

10.1. Client Behavior

A DHCPv4 client MAY emit a DHCPv4 option and a DHCPv6 client MAY emit DHCPv6 option. These options are singletons, as specified in [RFC7227]. Because clients are intended to have at most one MUD URL associated with them, they may emit at most one MUD URL option via DHCPv4 and one MUD URL option via DHCPv6. In the case where both v4 and v6 DHCP options are emitted, the same URL MUST be used.

10.2. Server Behavior

A DHCP server may ignore these options or take action based on receipt of these options. When a server consumes this option, it will either forward the URL and relevant client information (such as the gateway address or giaddr and requested IP address, and lease length) to a network management system, or it will retrieve the usage description itself by resolving the URL.

DHCP servers may implement MUD functionality themselves or they may pass along appropriate information to a network management system or MUD manager. A DHCP server that does process the MUD URL MUST adhere to the process specified in [RFC2818] and [RFC5280] to validate the TLS certificate of the web server hosting the MUD file. Those servers will retrieve the file, process it, create and install the necessary configuration on the relevant network element. Servers SHOULD monitor the gateway for state changes on a given interface. A DHCP server that does not provide MUD functionality and has forwarded a MUD URL to a MUD manager MUST notify the MUD manager of any corresponding change to the DHCP state of the client (such as expiration or explicit release of a network address lease).

Should the DHCP server fail, in the case when it implements the MUD manager functionality, any backup mechanisms SHOULD include the MUD state, and the server SHOULD resolve the status of clients upon its restart, similar to what it would do, absent MUD manager functionality. In the case where the DHCP server forwards information to the MUD manager, the MUD manager will either make use of redundant DHCP servers for information, or otherwise clear state based on other network information, such as monitoring port status on a switch via SNMP, Radius accounting, or similar mechanisms.

10.3. Relay Requirements

There are no additional requirements for relays.

11. The Manufacturer Usage Description (MUD) URL X.509 Extension

This section defines an X.509 non-critical certificate extension that contains a single Uniform Resource Locator (URL) that points to an on-line Manufacturer Usage Description concerning the certificate subject. URI must be represented as described in Section 7.4 of [RFC5280].

Any Internationalized Resource Identifiers (IRIs) MUST be mapped to URIs as specified in Section 3.1 of [RFC3987] before they are placed in the certificate extension.

The semantics of the URL are defined Section 6 of this document.

The choice of id-pe is based on guidance found in Section 4.2.2 of [RFC5280]:

> These extensions may be used to direct applications to on-line information about the issuer or the subject.

The MUD URL is precisely that: online information about the particular subject.

In addition, a separate new extension is defined as id-pe-mudsigner. This contains the subject field of the signing certificate of the MUD file. Processing of this field is specified in Section 13.2.

The purpose of this signature is to make a claim that the MUD file found on the server is valid for a given device, independent of any other factors. There are several security considerations below in Section 16.

A new content-type id-ct-mud is also defined. While signatures are detached today, should a MUD file be transmitted as part of a CMS message, this content-type SHOULD be used.

The new extension is identified as follows:

```
<CODE BEGINS>
   MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
                internet(1) security(5) mechanisms(5) pkix(7)
                id-mod(0) id-mod-mudURLExtn2016(88) }
    DEFINITIONS IMPLICIT TAGS ::= BEGIN
```

```
      -- EXPORTS ALL --

     IMPORTS

       -- RFC 5912
       EXTENSION
       FROM PKIX-CommonTypes-2009
            { iso(1) identified-organization(3) dod(6) internet(1)
              security(5) mechanisms(5) pkix(7) id-mod(0)
              id-mod-pkixCommon-02(57) }

       -- RFC 5912
       id-ct
       FROM PKIXCRMF-2009
            { iso(1) identified-organization(3) dod(6) internet(1)
              security(5)  mechanisms(5) pkix(7) id-mod(0)
              id-mod-crmf2005-02(55) }

       -- RFC 6268
       CONTENT-TYPE
       FROM CryptographicMessageSyntax-2010
         { iso(1) member-body(2) us(840) rsadsi(113549)
            pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

       -- RFC 5912
       id-pe, Name
       FROM PKIX1Explicit-2009
             { iso(1) identified-organization(3) dod(6) internet(1)
               security(5) mechanisms(5) pkix(7) id-mod(0)
               id-mod-pkix1-explicit-02(51) } ;

      --
      -- Certificate Extensions
      --

      MUDCertExtensions EXTENSION ::=
         { ext-MUDURL | ext-MUDsigner, ... }

      ext-MUDURL EXTENSION ::=
         { SYNTAX MUDURLSyntax IDENTIFIED BY id-pe-mud-url }

      id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }

      MUDURLSyntax ::= IA5String

      ext-MUDsigner EXTENSION ::=
         { SYNTAX MUDsignerSyntax IDENTIFIED BY id-pe-mudsigner }
```

```
      id-pe-mudsigner OBJECT IDENTIFIER ::= { id-pe TBD1 }

      MUDsignerSyntax ::= Name

      --
      -- CMS Content Types
      --

      MUDContentTypes CONTENT-TYPE ::=
         { ct-mud, ... }

       ct-mud CONTENT-TYPE ::=
         { -- directly include the content
           IDENTIFIED BY id-ct-mudtype }
         -- The binary data that is in the form
         -- 'application/mud+json" is directly encoded as the
         -- signed data.  No additional ASN.1 encoding is added.

      id-ct-mudtype OBJECT IDENTIFIER ::= { id-ct TBD2 }

      END
  <CODE ENDS>
```

While this extension can appear in either an 802.AR manufacturer certificate (IDevID) or deployment certificate (LDevID), of course it is not guaranteed in either, nor is it guaranteed to be carried over. It is RECOMMENDED that MUD manager implementations maintain a table that maps a Thing to its MUD URL based on IDevIDs.

12. The Manufacturer Usage Description LLDP extension

The IEEE802.1AB Link Layer Discovery Protocol (LLDP) is a one hop vendor-neutral link layer protocol used by end hosts network Things for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network. Its Type-Length-Value (TLV) design allows for 'vendor-specific' extensions to be defined. IANA has a registered IEEE 802 organizationally unique identifier (OUI) defined as documented in [RFC7042]. The MUD LLDP extension uses a subtype defined in this document to carry the MUD URL.

The LLDP vendor specific frame has the following format:

```
+--------+--------+----------+---------+--------------
|TLV Type|  len   |   OUI    |subtype  | MUDString
|  =127  |        |= 00 00 5E|  = 1    |
|(7 bits)|(9 bits)|(3 octets)|(1 octet)|(1-255 octets)
+--------+--------+----------+---------+--------------
```

where:

- o TLV Type = 127 indicates a vendor-specific TLV
- o len - indicates the TLV string length
- o OUI = 00 00 5E is the organizationally unique identifier of IANA
- o subtype = 1 (to be assigned by IANA for the MUD URL)
- o MUD URL - the length MUST NOT exceed 255 octets

The intent of this extension is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview of the network system as managed by the network administrator to decide what to do with this information. The key function of this extension is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

Hosts, routers, or other network elements that implement this option are intended to have at most one MUD URL associated with them, so they may transmit at most one MUD URL value.

Hosts, routers, or other network elements that implement this option may ignore these options or take action based on receipt of these options. For example they may fill in information in the respective extensions of the LLDP Management Information Base (LLDP MIB). LLDP operates in a one-way direction. LLDPDUs are not exchanged as information requests by one Thing and response sent by another Thing. The other Things do not acknowledge LLDP information received from a Thing. No specific network behavior is guaranteed. When a Thing consumes this extension, it may either forward the URL and relevant remote Thing information to a MUD manager, or it will retrieve the usage description by resolving the URL in accordance with normal HTTP semantics.

13. Creating and Processing of Signed MUD Files

Because MUD files contain information that may be used to configure network access lists, they are sensitive. To ensure that they have not been tampered with, it is important that they be signed. We make use of DER-encoded Cryptographic Message Syntax (CMS) [RFC5652] for this purpose.

13.1. Creating a MUD file signature

A MUD file MUST be signed using CMS as an opaque binary object. In order to make successful verification more likely, intermediate certificates SHOULD be included. The signature is stored at the location specified in the MUD file. Signatures are transferred using content-type "application/pkcs7-signature".

For example:

```
% openssl cms -sign -signer mancertfile -inkey mankey \
              -in mudfile -binary -outform DER -binary \
              -certfile intermediatecert -out mudfile.p7s
```

Note: A MUD file may need to be re-signed if the signature expires.

13.2. Verifying a MUD file signature

Prior to processing the rest of a MUD file, the MUD manager MUST retrieve the MUD signature file by retrieving the value of "mud-signature" and validating the signature across the MUD file. The Key Usage Extension in the signing certificate MUST be present and have the bit digitalSignature(0) set. When the id-pe-mudsigner extension is present in a device's X.509 certificate, the MUD signature file MUST have been generated by a certificate whose subject matches the contents of that id-pe-mudsigner extension. If these conditions are not met, or if it cannot validate the chain of trust to a known trust anchor, the MUD manager MUST cease processing the MUD file until an administrator has given approval.

The purpose of the signature on the file is to assign accountability to an entity, whose reputation can be used to guide administrators on whether or not to accept a given MUD file. It is already common place to check web reputation on the location of a server on which a file resides. While it is likely that the manufacturer will be the signer of the file, this is not strictly necessary, and may not be desirable. For one thing, in some environments, integrators may install their own certificates. For another, what is more important is the accountability of the recommendation, and not just the relationship between the Thing and the file.

An example:

```
% openssl cms -verify -in mudfile.p7s -inform DER -content mudfile
```

Note the additional step of verifying the common trust root.

14. Extensibility

One of our design goals is to see that MUD files are able to be understood by as broad a cross-section of systems as is possible. Coupled with the fact that we have also chosen to leverage existing mechanisms, we are left with no ability to negotiate extensions and a limited desire for those extensions in any event. A such, a two-tier extensibility framework is employed, as follows:

1. At a coarse grain, a protocol version is included in a MUD URL. This memo specifies MUD version 1. Any and all changes are entertained when this version is bumped. Transition approaches between versions would be a matter for discussion in future versions.

2. At a finer grain, only extensions that would not incur additional risk to the Thing are permitted. Specifically, adding nodes to the mud container is permitted with the understanding that such additions will be ignored by unaware implementations. Any such extensions SHALL be standardized through the IETF process, and MUST be named in the "extensions" list. MUD managers MUST ignore YANG nodes they do not understand and SHOULD create an exception to be resolved by an administrator, so as to avoid any policy inconsistencies.

15. Deployment Considerations

Because MUD consists of a number of architectural building blocks, it is possible to assemble different deployment scenarios. One key aspect is where to place policy enforcement. In order to protect the Thing from other Things within a local deployment, policy can be enforced on the nearest switch or access point. In order to limit unwanted traffic within a network, it may also be advisable to enforce policy as close to the Internet as possible. In some circumstances, policy enforcement may not be available at the closest hop. At that point, the risk of lateral infection (infection of devices that reside near one another) is increased to the number of Things that are able to communicate without protection.

A caution about some of the classes: admission of a Thing into the "manufacturer" and "same-manufacturer" class may have impact on access of other Things. Put another way, the admission may grow the access-list on switches connected to other Things, depending on how access is managed. Some care should be given on managing that access-list growth. Alternative methods such as additional network segmentation can be used to keep that growth within reason.

Because as of this writing MUD is a new concept, one can expect a great many devices to not have implemented it. It remains a local deployment decision as to whether a device that is first connected should be allowed broad or limited access. Furthermore, as mentioned in the introduction, a deployment may choose to ignore a MUD policy in its entirety, but simply taken into account the MUD URL as a classifier to be used as part of a local policy decision.

Finally, please see directly below regarding device lifetimes and use of domain names.

16. Security Considerations

Based on how a MUD URL is emitted, a Thing may be able to lie about what it is, thus gaining additional network access. This can happen in a number of ways when a device emits a MUD URL using DHCP or LLDP, such as being inappropriately admitted to a class such as "same-manufacturer", given access to a device such as "my-controller", or being permitted access to an Internet resource, where such access would otherwise be disallowed. Whether that is the case will depend on the deployment. Implementations SHOULD be configurable to disallow additive access for devices using MUD-URLs that are not emitted in a secure fashion such as in a certificate. Similarly, implementations SHOULD NOT grant elevated permissions (beyond those of devices presenting no MUD policy) to devices which do not strongly bind their identity to their L2/L3 transmissions. When insecure methods are used by the MUD Manager, the classes SHOULD NOT contain devices that use both insecure and secure methods, in order to prevent privilege escalation attacks, and MUST NOT contain devices with the same MUD-URL that are derived from both strong and weak authentication methods.

Devices may forge source (L2/L3) information. Deployments should apply appropriate protections to bind communications to the authentication that has taken place. For 802.1X authentication, IEEE 802.1AE (MACsec) [IEEE8021AE] is one means by which this may happen. A similar approach can be used with 802.11i (WPA2) [IEEE80211i]. Other means are available with other lower layer technologies. Implementations using session-oriented access that is not cryptographically bound should take care to remove state when any form of break in the session is detected.

A rogue CA may sign a certificate that contains the same subject name as is listed in the MUDsigner field in the manufacturer certificate, thus seemingly permitting a substitute MUD file for a device. There are two mitigations available: first, if the signer changes, this may be flagged as an exception by the MUD manager. If the MUD file also changes, the MUD manager SHOULD seek administrator approval (it should do this in any case). In all circumstances, the MUD manager MUST maintain a cache of trusted CAs for this purpose. When such a rogue is discovered, it SHOULD be removed.

Additional mitigations are described below.

When certificates are not present, Things claiming to be of a certain manufacturer SHOULD NOT be included in that manufacturer grouping without additional validation of some form. This will be relevant whenthe MUD manager makes use of primitives such as "manufacturer" for the purpose of accessing Things of a particular type. Similarly, network management systems may be able to fingerprint the Thing. In such cases, the MUD URL can act as a classifier that can be proven or disproven. Fingerprinting may have other advantages as well: when 802.1AR certificates are used, because they themselves cannot change, fingerprinting offers the opportunity to add artifacts to the MUD string in the form of the reserved field discussed in Section 10. The meaning of such artifacts is left as future work.

MUD managers SHOULD NOT accept a usage description for a Thing with the same MAC address that has indicated a change of the URL authority without some additional validation (such as review by a network administrator). New Things that present some form of unauthenticated MUD URL SHOULD be validated by some external means when they would be be given increased network access.

It may be possible for a rogue manufacturer to inappropriately exercise the MUD file parser, in order to exploit a vulnerability. There are three recommended approaches to address this threat. The first is to validate that the signer of the MUD file is known to and trusted by the MUD manager. The second is to have a system do a primary scan of the file to ensure that it is both parseable and believable at some level. MUD files will likely be relatively small, to start with. The number of ACEs used by any given Thing should be relatively small as well. It may also be useful to limit retrieval of MUD URLs to only those sites that are known to have decent web or domain reputations.

Use of a URL necessitates the use of domain names. If a domain name changes ownership, the new owner of that domain may be able to provide MUD files that MUD managers would consider valid. There are a few approaches that can mitigate this attack. First, MUD managers SHOULD cache certificates used by the MUD file server. When a new certificate is retrieved for whatever reason, the MUD manager should check to see if ownership of the domain has changed. A fair programmatic approximation of this is when the name servers for the domain have changed. If the actual MUD file has changed, the MUD manager MAY check the WHOIS database to see if registration ownership of a domain has changed. If a change has occurred, or if for some reason it is not possible to determine whether ownership has changed, further review may be warranted. Note, this remediation does not take into account the case of a Thing that was produced long ago and only recently fielded, or the case where a new MUD manager has been installed.

The release of a MUD URL by a Thing reveals what the Thing is, and provides an attacker with guidance on what vulnerabilities may be present.

While the MUD URL itself is not intended to be unique to a specific Thing, the release of the URL may aid an observer in identifying individuals when combined with other information. This is a privacy consideration.

In addressing both of these concerns, implementors should take into account what other information they are advertising through mechanisms such as mDNS[RFC6872], how a Thing might otherwise be identified, perhaps through how it behaves when it is connected to the network, whether a Thing is intended to be used by individuals or carry personal identifying information, and then apply appropriate data minimization techniques. One approach is to make use of TEAP [RFC7170] as the means to share information with authorized components in the network. Network elements may also assist in limiting access to the MUD URL through the use of mechanisms such as DHCPv6-Shield [RFC7610].

There is the risk of the MUD manager itself being spied on to determine what things are connected to the network. To address this risk, MUD managers may choose to make use of TLS proxies that they trust that would aggregate other information.

Please note that the security considerations mentioned in Section 4.7 of [I-D.ietf-netmod-rfc6087bis] are not applicable in this case because the YANG serialization is not intended to be accessed via NETCONF. However, for those who try to instantiate this model in a network element via NETCONF, all objects in each model in this draft exhibit similar security characteristics as [I-D.ietf-netmod-acl-model]. The basic purpose of MUD is to configure access, and so by its very nature can be disruptive if used by unauthorized parties.

17. IANA Considerations

[There was originally a registry entry for .well-known suffixes. This has been removed from the draft and may be marked as deprecated in the registry. RFC Editor: please remove this comment.]

17.1. YANG Module Registrations

The following YANG modules are requested to be registered in the "IANA Module Names" registry:

The ietf-mud module:

- o Name: ietf-mud
- o URN: urn:ietf:params:xml:ns:yang:ietf-mud
- o Prefix: ietf-mud
- o Registrant conact: The IESG
- o Reference: [RFCXXXX]

The ietf-acldns module:

- o Name: ietf-acldns
- o URI: urn:ietf:params:xml:ns:yang:ietf-acldns
- o Prefix: ietf-acldns
- o Registrant: the IESG
- o Reference: [RFCXXXX]

17.2. DHCPv4 and DHCPv6 Options

The IANA has allocated option 161 in the Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) Parameters registry for the MUD DHCPv4 option, and option 112 for DHCPv6, as described in Section 10.

17.3. PKIX Extensions

IANA is kindly requested to make the following assignments for:

o The MUDURLExtnModule-2016 ASN.1 module in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0).

o id-pe-mud-url object identifier from the "SMI Security for PKIX Certificate Extension" registry (1.3.6.1.5.5.7.1).

o id-pe-mudsigner object identifier from the "SMI Security for PKIX Certificate Extension" registry (TBD1).

o id-ct-mudtype object identifier from the "SMI Security for S/MIME CMS Content Type" registry (TBD2).

The use of these values is specified in Section 11.

17.4. MIME Media-type Registration for MUD files

The following media-type is defined for transfer of MUD file:

- o Type name: application
- o Subtype name: mud+json
- o Required parameters: n/a
- o Optional parameters: n/a
- o Encoding considerations: 8bit; application/mud+json values are represented as a JSON object; UTF-8 encoding MUST be employed. [RFC3629]
- o Security considerations: See Security Considerations of RFCXXXX and [RFC8259] Section 12.
- o Interoperability considerations: n/a
- o Published specification: [RFCXXXX]
- o Applications that use this media type: MUD managers as specified by [RFCXXXX].
- o Fragment identifier considerations: n/a
- o Additional information:

  Magic number(s): n/a
  File extension(s): n/a
  Macintosh file type code(s): n/a

- o Person & email address to contact for further information: Eliot Lear <lear@cisco.com>, Ralph Droms <rdroms@gmail.com>
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author:
  Eliot Lear <lear@cisco.com>
  Ralph Droms <rdroms@gmail.com>
- o Change controller: IESG
- o Provisional registration? (standards tree only): No.

17.5. LLDP IANA TLV Subtype Registry

IANA is requested to create a new registry for IANA Link Layer Discovery Protocol (LLDP) TLV subtype values. The recommended policy for this registry is Expert Review. The maximum number of entries in the registry is 256.

IANA is required to populate the initial registry with the value:

LLDP subtype value = 1 (All the other 255 values should be initially marked as ’Unassigned’.)

Description = the Manufacturer Usage Description (MUD) Uniform Resource Locator (URL)

Reference = < this document >

17.6. The MUD Well Known Universal Resource Name (URNs)

The following parameter registry is requested to be added in accordance with [RFC3553]

```
Registry name: "urn:ietf:params:mud" is requested.
Specification: this document
Repository: this document
Index value:  Encoded identically to a TCP/UDP port service
              name, as specified in Section 5.1 of [RFC6335]
```

The following entries should be added to the "urn:ietf:params:mud" name space:

"urn:ietf:params:mud:dns" refers to the service specified by [RFC1123]. "urn:ietf:params:mud:ntp" refers to the service specified by [RFC5905].

17.7. Extensions Registry

The IANA is requested to establish a registry of extensions as follows:

```
Registry name: MUD extensions registry
Registry policy: Standards action
Standard reference: document
Extension name: UTF-8 encoded string, not to exceed 40 characters.
```

Each extension MUST follow the rules specified in this specification. As is usual, the IANA issues early allocations based in accordance with [RFC7120].

18. Acknowledgments

The authors would like to thank Einar Nilsen-Nygaard, who singlehandedly updated the model to match the updated ACL model, Bernie Volz, Tom Gindin, Brian Weis, Sandeep Kumar, Thorsten Dahm, John Bashinski, Steve Rich, Jim Bieda, Dan Wing, Joe Clarke, Henk Birkholz, Adam Montville, Jim Schaad, and Robert Sparks for their valuable advice and reviews. Russ Housley entirely rewrote

Section 11 to be a complete module. Adrian Farrel provided the basis for privacy considerations text. Kent Watsen provided a thorough review of the architecture and the YANG model. The remaining errors in this work are entirely the responsibility of the authors.

19. References

19.1. Normative References

[I-D.ietf-netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-19 (work in progress), April 2018.

[IEEE8021AB]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery", n.d..

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://www.rfc-editor.org/info/rfc2131>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <https://www.rfc-editor.org/info/rfc3315>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/info/rfc3987>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/info/rfc5911>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <https://www.rfc-editor.org/info/rfc7227>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7610] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers", BCP 199, RFC 7610, DOI 10.17487/RFC7610, August 2015, <https://www.rfc-editor.org/info/rfc7610>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <https://www.rfc-editor.org/info/rfc8348>.

19.2. Informative References

[FW95] Chapman, D. and E. Zwicky, "Building Internet Firewalls", January 1995.

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20 (work in progress), March 2018.

[IEEE80211i] Institute for Electrical and Electronics Engineers, "IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11- Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications- Amendment 6- Medium Access Control (MAC) Security Enhancements", 2004.

[IEEE8021AE] Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks- Media Access Control (MAC) Security", 2006.

[IEEE8021AR] Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.

[IEEE8021X] Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", 2010.

[ISO.8601.1988]
International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, June 1988.

[RFC1984] IAB and IESG, "IAB and IESG Statement on Cryptographic Technology and the Internet", BCP 200, RFC 1984, DOI 10.17487/RFC1984, August 1996, <https://www.rfc-editor.org/info/rfc1984>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <https://www.rfc-editor.org/info/rfc3553>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <https://www.rfc-editor.org/info/rfc6092>.

[RFC6872] Gurbani, V., Ed., Burger, E., Ed., Anjali, T., Abdelnur, H., and O. Festor, "The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Information Model", RFC 6872, DOI 10.17487/RFC6872, February 2013, <https://www.rfc-editor.org/info/rfc6872>.

[RFC7042] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, <https://www.rfc-editor.org/info/rfc7042>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <https://www.rfc-editor.org/info/rfc7170>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <https://www.rfc-editor.org/info/rfc7452>.

[RFC7488] Boucadair, M., Penno, R., Wing, D., Patil, P., and T. Reddy, "Port Control Protocol (PCP) Server Selection", RFC 7488, DOI 10.17487/RFC7488, March 2015, <https://www.rfc-editor.org/info/rfc7488>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

Appendix A. Changes from Earlier Versions

RFC Editor to remove this section prior to publication.

Draft -19: * Edits after discussion with apps area to address reserved field for the future. * Correct systeminfo to be utf8. * Remove "hardware-rev" from list.

Draft -18: * Correct an error in the augment statement * Changes to the ACL model re ports.

Draft -17:

o One editorial.

Draft -16

o add mud-signature element based on review comments

o redo mud-url

o make clear that systeminfo uses UTF8

Draft -13 to -14:

o Final WGLC comments and review comments

o Move version from MUD-URL to Model

o Have MUD-URL in model

o Update based on update to draft-ietf-netmod-acl-model

o Point to tree diagram draft instead of 6087bis.

Draft -12 to -13:

- o Additional WGLC comments

Draft -10 to -12:

These are based on WGLC comments:

- o Correct examples based on ACL model changes.
- o Change ordering nodes.
- o Additional explanatory text around systeminfo.
- o Change ordering in examples.
- o Make it VERY VERY VERY VERY clear that these are recommendations, not mandates.
- o DHCP -> NTP in some of the intro text.
- o Remove masa-server
- o "Things" to "network elements" in a few key places.
- o Reference to JSON YANG RFC added.

Draft -10 to -11:

- o Example corrections
- o Typo
- o Fix two lists.
- o Addition of ’any-acl’ and ’mud-acl’ in the list of allowed features.
- o Clarification of what should be in a MUD file.

Draft -09 to -10:

- o AD input.
- o Correct dates.
- o Add compliance sentence as to which ACL module features are implemented.

Draft -08 to -09:

- o Resolution of Security Area review, IoT directorate review, GenART review, YANG doctors review.
- o change of YANG structure to address mandatory nodes.
- o Terminology cleanup.
- o specify out extra portion of MUD-URL.
- o consistency changes.
- o improved YANG descriptions.
- o Remove extra revisions.
- o Track ACL model changes.
- o Additional cautions on use of ACL model; further clarifications on extensions.

Draft -07 to -08:

- o a number of editorials corrected.
- o definition of MUD file tweaked.

Draft -06 to -07:

- o Examples updated.
- o Additional clarification for direction-initiated.
- o Additional implementation guidance given.

Draft -06 to -07:

- o Update models to match new ACL model
- o extract directionality from the ACL, introducing a new device container.

Draft -05 to -06:

- o Make clear that this is a component architecture (Polk and Watson)
- o Add order of operations (Watson)

o Add extensions leaf-list (Pritikin)

o Remove previous-mud-file (Watson)

o Modify text in last-update (Watson)

o Clarify local networks (Weis, Watson)

o Fix contact info (Watson)

o Terminology clarification (Weis)

o Advice on how to handle LDevIDs (Watson)

o Add deployment considerations (Watson)

o Add some additional text about fingerprinting (Watson)

o Appropriate references to 6087bis (Watson)

o Change systeminfo to a URL to be referenced (Lear)

Draft -04 to -05: * syntax error correction

Draft -03 to -04: * Re-add my-controller

Draft -02 to -03: * Additional IANA updates * Format correction in YANG. * Add reference to TEAP.

Draft -01 to -02: * Update IANA considerations * Accept Russ Housley rewrite of X.509 text * Include privacy considerations text * Redo the URL limit. Still 255 bytes, but now stated in the URL definition. * Change URI registration to be under urn:ietf:params

Draft -00 to -01: * Fix cert trust text. * change supportInformation to meta-info * Add an informational element in. * add urn registry and create first entry * add default elements

Appendix B. Default MUD nodes

What follows is the portion of a MUD file that permits DNS traffic to a controller that is registered with the URN "urn:ietf:params:mud:dns" and traffic NTP to a controller that is registered "urn:ietf:params:mud:ntp". This is considered the default behavior and the ACEs are in effect appended to whatever other "ace" entries that a MUD file contains. To block DNS or NTP one repeats the matching statement but replaces the "forwarding" action "accept" with "drop". Because ACEs are processed in the order they are received, the defaults would not be reached. A MUD manager might further decide to optimize to simply not include the defaults when they are overridden.

Four "acl" list entries that implement default MUD nodes are listed below. Two are for IPv4 and two are for IPv6 (one in each direction for both versions of IP). Note that neither access-list name nor ace name need be retained or used in any way by local implementations, but are simply there for completeness' sake.


```
 "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-59776-v4to",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "ent0-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "urn:ietf:params:mud:dns"
                },
                "ipv4": {
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            },
            {
              "name": "ent1-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "urn:ietf:params:mud:ntp"
                },
                "ipv4": {
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
```

```
                       "operator": "eq",
                       "port": 123
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         },
         {
           "name": "mud-59776-v4fr",
           "type": "ipv4-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "ent0-frdev",
                 "matches": {
                   "ietf-mud:mud": {
                     "controller": "urn:ietf:params:mud:dns"
                   },
                   "ipv4": {
                     "protocol": 17
                   },
                   "udp": {
                     "destination-port": {
                       "operator": "eq",
                       "port": 53
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               },
               {
                 "name": "ent1-frdev",
                 "matches": {
                   "ietf-mud:mud": {
                     "controller": "urn:ietf:params:mud:ntp"
                   },
                   "ipv4": {
                     "protocol": 17
                   },
                   "udp": {
                     "destination-port": {
```

```
                      "operator": "eq",
                      "port": 123
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "mud-59776-v6to",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "ent0-todev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "source-port": {
                      "operator": "eq",
                      "port": 53
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              },
              {
                "name": "ent1-todev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "source-port": {
```

```
                      "operator": "eq",
                      "port": 123
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "mud-59776-v6fr",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "ent0-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
                      "operator": "eq",
                      "port": 53
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              },
              {
                "name": "ent1-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
```

```
                       "operator": "eq",
                       "port": 123
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         }
       ]
     }
```


Appendix C. A Sample Extension: DETNET-indicator

In this sample extension we augment the core MUD model to indicate whether the device implements DETNET. If a device claims not to use DETNET, but then later attempts to do so, a notification or exception might be generated. Note that this example is intended only for illustrative purposes.

Extension Name: "Example-Extension" (to be used in the extensions list)
Standard: this document (but do not register the example)

This extension augments the MUD model to include a single node, using the following sample module that has the following tree structure:

```
module: ietf-mud-detext-example
  augment /ietf-mud:mud:
    +--rw is-detnet-required?   boolean
```

The model is defined as follows:

```
<CODE BEGINS>file "ietf-mud-detext-example@2018-06-15.yang"
module ietf-mud-detext-example {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-detext-example";
  prefix ietf-mud-detext-example;

  import ietf-mud {
    prefix ietf-mud;
  }
```

```
     organization
       "IETF OPSAWG (Ops Area) Working Group";
     contact
       "WG Web: http://tools.ietf.org/wg/opsawg/
        WG List: opsawg@ietf.org
        Author: Eliot Lear
        lear@cisco.com
        Author: Ralph Droms
        rdroms@gmail.com
        Author: Dan Romascanu
        dromasca@gmail.com

       ";
     description
       "Sample extension to a MUD module to indicate a need
        for DETNET support.";

     revision 2018-06-15 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: Manufacturer Usage Description
          Specification";
     }

     augment "/ietf-mud:mud" {
       description
         "This adds a simple extension for a manufacturer
            to indicate whether DETNET is required by a
          device.";
       leaf is-detnet-required {
         type boolean;
         description
           "This value will equal true if a device requires
            detnet to properly function";
       }
     }
   }
   <CODE ENDS>
```

Using the previous example, we now show how the extension would be expressed:

```
   {
     "ietf-mud:mud": {
       "mud-version": 1,
       "mud-url": "https://lighting.example.com/lightbulb2000",
       "last-update": "2018-03-02T11:20:51+01:00",
```

```
       "cache-validity": 48,
       "extensions": [
           "ietf-mud-detext-example"
        ],
       "ietf-mud-detext-example:is-detnet-required": "false",
       "is-supported": true,
       "systeminfo": "The BMS Example Lightbulb",
       "from-device-policy": {
         "access-lists": {
           "access-list": [
             {
               "name": "mud-76100-v6fr"
             }
           ]
         }
       },
       "to-device-policy": {
         "access-lists": {
           "access-list": [
             {
               "name": "mud-76100-v6to"
             }
           ]
         }
       }
     },
     "ietf-access-control-list:acls": {
       "acl": [
         {
           "name": "mud-76100-v6to",
           "type": "ipv6-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "cl0-todev",
                 "matches": {
                   "ipv6": {
                     "ietf-acldns:src-dnsname": "test.example.com",
                     "protocol": 6
                   },
                   "tcp": {
                     "ietf-mud:direction-initiated": "from-device",
                     "source-port": {
                       "operator": "eq",
                       "port": 443
                     }
                   }
                 },
```

```
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         },
         {
           "name": "mud-76100-v6fr",
           "type": "ipv6-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "cl0-frdev",
                 "matches": {
                   "ipv6": {
                     "ietf-acldns:dst-dnsname": "test.example.com",
                     "protocol": 6
                   },
                   "tcp": {
                     "ietf-mud:direction-initiated": "from-device",
                     "destination-port": {
                       "operator": "eq",
                       "port": 443
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "accept"
                 }
               }
             ]
           }
         }
       ]
     }
   }
```


Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Ralph Droms
Google
355 Main St., 5th Floor
Cambridge

Phone: +1 978 376 3731
Email: rdroms@gmail.com

Dan Romascanu

Phone: +972 54 5555347
Email: dromasca@gmail.com

Network Working Group Z. Li
Internet-Draft Huawei Technologies
Intended status: Informational Y. Zheng
Expires: September 14, 2017 China Unicom
J. Zhang
S. Xu
Huawei Technologies
March 13, 2017

An Architecture of Network Artificial Intelligence(NAI)
draft-li-opsawg-network-ai-arch-00

Abstract

Artificial intelligence is an important technical trend in the industry. With the development of network, it is necessary to introduce artificial intelligence technology to achieve self-adjustment, self- optimization, self-recovery of the network through collection of huge data of network state and machine learning. This draft defines the architecture of Network Artificial Intelligence (NAI), including the key components and the key protocol extension requirements.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described inRFC 2119 [RFC2119]

Copyright Notice

Table of Contents

1. Introduction

Artificial Intelligence is an important technical trend in the industry. The two key aspects of Artificial Intelligence are perception and cognition. Artificial Intelligence has evolved from an early non-learning expert system to a learning-capable machine learning era. In recent years, the rapid development of the deep learning branch based on the neural network and the maturity of the big data technology and software distributed architecture make the Artificial Intelligence in many fields (such as transportation, medical treatment, education, etc.) have been applied. With the development of network, it is necessary to introduce artificial intelligence technology to achieve self-adjustment, self-optimization, self-recovery of the network through collection of huge data of network state and machine learning. The areas of machine learning which are easier to be used in the network field may include: root cause analysis of network failures, network traffic prediction, traffic adjustment and optimization, security defense, security auditing, etc., to implement network perception and cognition.

This draft defines the architecture of Network Artificial Intelligence (NAI), including the key components and the key protocol extension requirements.

2. Terminology

AI: Artificial Intelligence

NAI: Network Artificial Intelligence

3. Architecture

```
                   ^                                   ^
                (4)|                                   |(4)
   +---------------|--------------+    +---------------|--------------+
   | Domain 1      |              |    |               |     Domain 2 |
   |        +------------+        |    |        +------------+        |
   |        |  Central   |        |    |        |  Central   |        |
   |     (1)| Controller |----------------------| Controller |(1)     |
   |        |    with    |        |    |        |    with    |        |
   |        |     NTA    |        |    |        |     NTA    |        |
   |        +------------+        |    |        +------------+        |
   |         /          \         |    |         /          \         |
   |     (3)/            \        |    |        /            \(3)     |
   |       /              \       |    |       /              \       |
   | +--------+        +--------+ |    | +--------+        +--------+ |
   | |        |        |        | |    | |        |        |        | |
   | |Network | ...... |Network | |    | |Network | ...... |Network | |
   | | Device |  (2)   | Device | |    | | Device |  (2)   | Device | |
   | |    1   |        |    N   | |    | |    1   |        |    N   | |
   | +--------+        +--------+ |    | +--------+        +--------+ |
   |                              |    |                              |
   +------------------------------+    +------------------------------+
```

Figure 1: An Architecture of Network Artificial Intelligence(NAI)

The architecture of Network artificial intelligence includes following key components:

(1) Central Controller: Centralized controller is the core part of Network Artificial Intelligence which can be called as 'Network

Brain’. The Network Telemetry and Analytics (NTA) engines can be introduced acompanying with the central controller. The Network Telemetry and Analytics (NTA) engine inclues data collector, analytics framework, data persistence, and NAI applications.

(2) Network Device: IP network operation and maintenance are always a big challenge since the network can only provide limited state information. The network states includes but are not limited to topology, traffic engineering, operation and maintenance information, network failure information and related information to locate the network failure. In order to provide these information, the network must be able to support more OAM mechanisms to acquire more state information and report to the controller. Then the controller can get the complete state information of the network which is the base of Network Artificial Intelligence(NAI).

(3) Southbound Protocol and Models of Controller: As network devices provide huge network state information, it proposes a number of new requirements for protocols and models between controllers and network devices. The traditional southbound protocol such as Netconf and SNMP can not meet the performance requirements. It is necessary to introduce some new high-performance protocols to collect network state data. At the same time, the models of network data should be completed. Moreover with the introduction of new OAM mechanisms of network devices, new models of network data should be introduced.

(4) Northbound Model of Controller: The goal of the Network Artificial Intelligence is to reduce the technical requirements on the network administrators and release them from the heavy network management, control, maintenance work. The abstract northbound model of the controller for different network services should be simple and easy to be understood.

4. Process

NAI consists of following processes:

-- Data Collection

From the time aspect, data collection can be divided into real-time data collection and non-real-time collection.

From the content aspect, data collection can be divided into network information collection (including topology, tunnels, routing, equipment configuration, etc.) and traffic collection (the collection network traffic, network load, device KPI, etc.).

-- Data Storage

Store data collected from network. Many existing big data storage technologies can be used here.

-- Data Processing

This is preliminary data processing too select effective data and simply analyse data relationship.

-- Analyse

Analyse engine will provide the data analysis results using machine learning algorithm.

-- Closed Loop Control

According to the results of intelligent analysis and policy set by user, the central controller will implement closed-loop control of the network.

5. Classification

NAI can be divided into off-line process and on-line process in accordance to the time aspect of the data collection and analysis.

Off-line process refers to process of the existing data, or non-real-time collection data. Although the analysis process will also focus on the relationship between data and time, but it does not require real-time analysis. Off-line process is mainly used for two purposes: (1) training or verification of real-time process design; (2) trouble shooting or reason analysis for events that have already occurred.

On-line process is efficient real-time collection, processing and analysis of the data, to operate network monitoring and event forecasting. The main purpose of the on-line process are: (1) network capacity monitoring and precise optimizing; (2) network event prediction and fast trouble shooting; (3) real-time network optimization according to the policy.

6. Requirement of Protocol Extensions

6.1. Requirement of Southbound Protocols

REQ 01: The southbound protocol of the controller should be introduced to meet the performance requirements of collecting huge data of network states.

The soundbound protocol can be based on the extensions of the existing traditional protocols such link state colloction protocols, PCEP[RFC5440], BMP[RFC7854], etc. Or the new protocol like Telemetry[I-D.kumar-rtgwg-grpc-protocol] can be introduced as the soundbound protocols. The protocol choice will be based on the application scenarios of NAI.

6.2. Requirement of Data Collection

REQ 02: The data collected from the network devices includes but not limites to following information:

-- network topology information

-- routing protocol status

-- IP routes and MAC routes

-- LSP information

-- network traffic inforamtion

-- network configuration

-- network device KPIs

-- log of network elements

-- trap of network elements

-- OAM information

6.3. Requirement of Devices

REQ 03: New OAM mechanisms should be introduced for the network devices in order to acquire more types of network state data.

6.4. Requirement of Northbound Interface

REQ 04: The abstract network-based service models should be provided by the controller as the northbound models to satisfy the requirements of different services.

7. IANA Considerations

This document makes no request of IANA.

8. Security Considerations

TBD.

9. Normative References

[I-D.kumar-rtgwg-grpc-protocol]
Kumar, A., Kolhe, J., Ghemawat, S., and L. Ryan, "gRPC Protocol", draft-kumar-rtgwg-grpc-protocol-00 (work in progress), July 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <http://www.rfc-editor.org/info/rfc5440>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <http://www.rfc-editor.org/info/rfc7854>.

Authors' Addresses

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Yi Zheng
China Unicom
No.9, Shouti Nanlu, Haidian District
Beijing 100048
China

Email: zhengyi39@chinaunicom.cn

Jinhui Zhang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: jason.zhangjinhui@huawei.com

Xu Shiping
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
P.R. China

Email: xushiping7@huawei.com

Operations and Management Area Working Group — B. Pularikkal
Internet-Draft — Cisco Systems
Intended status: Informational — T. Pauly
Expires: January 4, 2018 — Apple Inc.
M. Grayson
S. Gundavelli
Cisco Systems
S. Touati
Ericsson
July 3, 2017

Carrier Wi-Fi Calling Deployment Considerations
draft-pularikkal-opsawg-wifi-calling-03

Abstract

Carrier Wi-Fi Calling is a solution that allows mobile operators to seamlessly offload mobile voice signaling and bearer traffic onto Wi-Fi access networks, which may or may not be managed by the mobile operators. Mobile data offload onto Wi-Fi access networks has already become very common, as Wi-Fi access has become more ubiquitous. However, the offload of mobile voice traffic onto Wi-Fi networks has become prevalent only in recent years. This was primarily driven by the native Wi-Fi Calling client support introduced by device vendors. The objective of this document is to provide a high level deployment reference to Mobile Operators and Wi-Fi Operators on Carrier Wi-Fi Calling.

Copyright Notice

Table of Contents

1. Introduction

There are several SP Managed and Over the Top Voice Solutions deployed today which can leverage Wi-Fi access networks. Some of these solutions rely on standalone applications installed on the Mobile Handset and other Mobile devices such as tablets. Also there are solutions, which leverage dedicated hardware built exclusively to support Voice over Wi-Fi.e.g,in enterprise type environments. The scope of this document is VoWiFi solutions, which are deployed by Mobile Network Operators also known as Wireless Carriers. VoWiFi from the context of Mobile Voice offload is often referred to as Carrier Wi-Fi Calling. The deployment of Carrier Wi-Fi Calling requires some kind of integration between the Wi-Fi Access network and Mobile Packet Core. Carrier Wi-Fi calling solutions deployed today predominantly uses an 'untrusted Wi-Fi'model that delivers simple IP connectivity to facilitate Mobile Packet Core integration. With this 'untrusted' approach, Mobile Operators are able to make use of the existing Wi-Fi deployment footprint regardless of whether it is owned by the MNOs or by their roaming partners or Wi-Fi Operators without any kind of partnership with the MNOs. This model has definitely allowed MNOs to accelerate the adoption of Wi-Fi calling. However, this comes with some caveats, as depending on the Wi-Fi network, there may be no visibility or control over it by the MNO, impacting its ability to carry voice calls without compromising end user experience.

It is in the interest of both MNOs as well as Wi-Fi Operators to improve the quality of experience for Wi-Fi Calling delivered over a Wi-Fi access network. MNOs have the incentive to make sure that the end user experience does not get compromised while the voice service is offloaded over Wi-Fi access. Wi-Fi operators have the business incentive to enter into roaming partnerships with the MNOs and support Wi-Fi calling with certain Service Level Agreements. In some deployments, it is possible for the MNOs to own some Wi-Fi hotspot deployments. In such cases, MNO will effectively be the Wi-Fi operator as well.

Objective of this document is to provide a Carrier Wi-Fi Calling deployment reference to Wi-Fi Operators and MNOs with primary focus on the Wi-Fi Access Network and the Wi-Fi to Packet Core integration aspects.

2. Terminology

Service Provider (SP)

Refers to a provider of telecommunications services such as Broadband Operator or Mobile Operator. An SP may provide several telecommunications services.

APP

Refers to computer program typically designed to run on Mobile devices such as smartphones and tablets.

Wireless Fidelity (Wi-Fi)

Technology that allows devices to wirelessly connect using 2.4 GHz and 5.0 GHz unlicensed radio bands. Wi-Fi is defined as part of IEEE 802.11 standards

Voice over Wi-Fi (VoWiFi)

Any solution, which supports voice services over Wi-Fi.

Mobile Network Operator (MNO)

A wireless communications service provider who owns and operates licensed wireless access network and the backend infrastructure to offer mobile voice, data and multimedia services.

3rd Generation Partnership Project (3GPP)

3GPP unites seven telecommunications standards development organizations known as Organizational Partners and provides their members with a stable environment to produce the reports and specifications that define 3GPP technologies

Groups Special Mobile Association (GSMA)

GSMA represents the interests of mobile operators worldwide, uniting nearly 800 operators with more than 250 companies in the broader mobile ecosystem, including handset and device makers, software companies, equipment providers and internet companies, as well as organizations in adjacent industry sectors.

User Equipment (UE)

Term represents any device used directly by an end user to communicate.

Wireless Local Area Network (WLAN)

Refers to IEEE 802.11 based Wi-Fi access networks and represents an extended service set consisting of multiple access points.

Long Term Evolution (LTE)

Is the fourth generation 3GPP standard set for wireless communication of mobile devices in end-to-end IP environment.

Evolved Packet Core (EPC)

Represents the Core Network in the 3GPP LTE system Architecture.

Packet Data Network (PDN)

PDN represents a network in the packet core a Mobile UE device wants to communicate with. PDN generally is mapped to a set of related services.

Access Point Name (APN)

APN represents a set of services available to a specific PDN. Typically UE devices will be configured to access multiple APNs corresponding various services in the packet core.

Trusted WLAN Access Gateway (TWAG)

Performs the gateway function between a trusted WLAN access network and packet core. It acts as the default gateway and DHCP Server for UE devices connected to the WLAN access network for trusted Wi-Fi to packet core integration model.

Evolved Packet Data Gateway (ePDG)

ePDG performs the gateway function between WLAN access network and Mobile Packet core in an untrusted model. Main function of ePDG is to secure the data transmission with a UE connected to the EPC.

PDN Gateway (P-GW)

P-GW is the subscriber session anchor in EPC. It enforces policy and also has a role in IP persistence in roaming scenarios. Based up on the policy, P-GW steers traffic towards various PDN networks corresponding to various APNs.

IP Multi-Media Subsystem (IMS)

An Architectural framework for delivering IP multimedia services. And is defined in 3GPP

Policy and Charging Rule Function (PCRF)

A system in EPC, which detects service data flows, applies policies and QoS to subscriber flows to and supports flow based charging

Session Initiation Protocol (SIP)

SIP is an application layer control protocol that can establish, modify and terminate multimedia sessions or calls.

Real-time Transport Protocol (RTP)

RTP is a transport protocol, which provides end-to-end delivery services for data with real-time characteristics such as interactive audio and video.

Proxy Mobile IPv6 (PMIPv6)

PMIPv6 is a network based mobility management protocol standardized by IETF and adopted in 3GPP.

GPRS Tunneling Protocol (GTP)

Group of IP based communications protocols used in 3GPP architectures.

S2a Interface

Is the interface between TWAG and P-GW and can be either GTP or PMIPv6 based

S2b Interface

Interface between ePDG and P-GW and can be either GTP or PMIPv6

3. Architecture Overview

This section provides a very high level overview of the end-to-end Architecture for Carrier Wi-Fi Calling. It is outside the scope of this document to provide a detailed Architecture description, as all the functional entities and the protocol interfaces are well defined in the 3GPP and GSMA specifications [3GPPTS23.402,GSMAIR61,GSMAIR51]. Figure-01 below is used to describe the Architecture components at a high level.

```
                                          +-----+     +-----+
                                          | 3GPP+-----+ HSS |
                                          | AAA |     +-----+
                                          +-----+
                                             |
                                             |        +-----+
                                             |        | PCRF|
                                             |        +-----+
                                             |           |
                                         +--------+      |
        +---+   +-----+  Backhaul| TWAG /|    +-----+
        |UE +---+WLAN +----------+ ePDG  +----+ PGW |
        +---+   +-----+          |       |    +-----+
                                 +--------+      |
                                                 |
                                              +-----+
                                              | IMS |
                                              +-----+
```

Figure 1: High Level Architecture

The UE is the end user device such as a smartphone running native Wi-Fi Calling client. The UE is connected to a Wi-Fi access network, which is represented by the block WLAN in the diagram. Depending up on the trust model, TWAG or ePDG gateway is used to integrate the WLAN access network to the MNO packet core.More details around this untrusted and trusted approaches are covered in the next section. The P-GW acts as the common anchor for the subscriber sessions regardless of whether the UE is connected to Wi-Fi or LTE (not shown), allowing the preservation of the IP Session during a handover between LTE and Wi-Fi. IMS provides several functions related to SIP based call control signaling, namely SIP authentication, basic telephony services, supplementary services, interworking with other IMS systems, and offload into circuit switched voice networks. In addition to voice, the same IMS infrastructure may be leveraged for other multi-media functions such as video calling. The IMS framework consists of several functional entities and is omitted for the sake of simplicity here. PCRF performs classical Policy and Charging Rule functions in the Mobile Packet Core. For the Wi-Fi calling solution, it will trigger the establishment of the default and dedicated bearers on the S2a or S2b interfaces for SIP and RTP traffic between the PGW and the TWAG/ePDG.

4. Wi-Fi Calling Deployment Considerations

This section covers deployment considerations for an end-to-end Wi-Fi calling Architecture that can influence the quality of experience, availability and monetization aspects of the solution offering.

4.1. Wi-Fi to Packet Core Integration

There are three different Architecture options available for Wi-Fi to Packet Core integration for the deployment of Wi-Fi calling. Each of these models are described in the sub-sections below:

4.1.1. Untrusted Model

This model is built around the assumption that the Wi-Fi access network is 'unmanaged' or untrusted from the MNOs perspective. Since this model does not rely on any security or data privacy implementations on the Wi-Fi access network, it requires the establishment of an IPSec tunnel between the UE device and the Mobile Packet Core. The ePDG gateway acts as the IPSec tunnel termination point on the packet core side. The ePDG handles the user authentication as well as the establishment of an S2b packet data network connection towards the P-GW using the GTP based S2b interface. This Architecture model is illustrated in figure-2 below.

```
      +--------------+            +----------+          +-------+
      | +----------+ |  IMS-APN   |          |          |       |
      | |    SWu   | |  Traffic   |          |          |       |
      | |  Client  +------------------------------------+       |
      | |          | | Other APN  |          |          | ePDG  |
      | |          | |  traffic   |          |          |       |
      | |          +------------------------------------+       |
      | |          | |            |          |          +-------+
      | +----------+ |            |          |            |  |
      |              |            |          |         S2b|  |S2b
      |     UE       |            |  WLAN    |            |  +---+
      |              |            | Access   |            |      |
      | +----------+ |  LBO       |          |    +-------+   +-------+
      | | Native   | |  Traffic   |          |    |IMS APN|   | Other |
      | |          | |            |          |    |  PGW  |   |  APN  |
      | | Client   +-------------------+     |    |       |   |  PGW  |
      | |          | |            |    |     |    +-------+   +-------+
      | +----------+ |            |    |     |         |          |
      +--------------+            +----------+         |          |
                                       |               |          |
                                       |           +-------+   +-------+
                                       |           |  IMS  |   |  App  |
                                       |           |       |   | Server|
                                       v           +-------+   +-------+
```

Figure 2: Untrusted Wi-Fi to Packet Core Integration Model for Wi-Fi Calling

The Wi-Fi calling client implementation uses the ePDG client for IMS APN while the default PDN or Internet APN traffic is locally offloaded (Local Breakout LBO) into the Wi-Fi access network. The "untrusted Wi-Fi" architecture supports multiple APN over SWu, allowing the MNO to also route specific applications traffic associated with one or more APN through the Packet Core, in addition to the IMS APN, if required.

4.1.1.1. IPSec Tunnel Negotation

The IPSec tunnel from the UE to the ePDG is negotiated using IKEv2. The parameters for tunnel negotation in Wi-Fi Calling are as follows:

- o The Initiator Identifier (IDi) will be in ID_RFC822_ADDR (email address) form, and be based on the UE’s IMSI@Realm.

- o The Responder Identifier (IDr) will be in ID_FQDN form, and be the APN name that the tunnel should access through the ePDG.

- o EAP should be used for mutual authentication. When on a device with a SIM card, EAP-AKA should be used. On other devices, EAP-TLS is preferred. EAP-Only authentication (in which the server certificate is not sent in an CERT payload) may be used to reduce packet size, but only with mutually authenticating EAP types such as EAP-AKA or EAP-TLS.

- o Strong encryption and authentication algorithms should be used, such as ENCR_AES_CBC, PRF_HMAC_SHA2_256, AUTH_HMAC_SHA2_256_128, and Diffie-Hellman Group 14.

- o The Configuration Request should specify an IPv4 or IPv6 addresses used for handover. The UE may also request ePDG-specific attributes such as P_CSCF_IP4_ADDRESS and P_CSCF_IP6_ADDRESS.

4.1.2. Hybrid Model

3GPP TS 23.402 also defines the concept of "trusted Wi-Fi" architecture, providing another method to integrate with the packet core. The trustworthiness of an access network itself is left to the MNO to decide, but it generally relies on some level of control by the MNO of the Wi-Fi access network either in a direct or indirect manner. One of the key characteristics of the "Trusted Wi-Fi" architecture as defined in 3GPP Release 11, is the client-less approach to support the packet core integration. This solution lacked the support for multiple APNs signaling for the UE when over the Wi-Fi access network, therefore all Wi-Fi offloaded traffic was assumed to be part of the default PDN or Internet APN. With this limitation, Wi-Fi calling cannot be supported as it require its own IMS APN. The hybrid architecture proposed here combines the 3GPP release 11 "trusted Wi-Fi" architecture, with the ePDG based untrusted Wi-Fi architecture. This hybrid model simultaneously supports IMS and other applications specific APNs using the untrusted Wi-Fi model, with the TWAG selectively offloading their traffic, while using the S2a interface for all other default PDN traffic toward the default PGW. This Architecture model is illustrated in figure 3 below

```
                                                  +------+   +---------+
                                                  | IMS  |   |  Other  |
                                                  | Core |   |Services |
                                                  +------+   +---------+
                                                     |           |
                                                  +------+   +-------+
       +--------------+      +-----------+        |IMS   |   | Other |
       | +----------+ |      |           |        |P-GW  |   | P-GW  |
       | |    SWu   | |      | +-------+ |        +------+   +-------+
       | |  Client  | |      | |SIPTO  | |           |           |
       | |          | |      | ++NAT   | |           |  +--------+
       | +----------+ |      | +-------+ |           |  |
       |              |      |           |        +------+
       |     UE       +------+   TWAG    |  S2b   | ePDG |
       |              |      |           +--------+      |
       | +----------+ |      |           |        +------+
       | | Native   | |      |           |
       | | Client   | |      |           |
       | |          | |      |           |  S2a   +-------+
       | +----------+ |      |           +--------+Default|
       +--------------+      +-----------+        | PGW   |
                                                  +-------+
                                                    |
                                                    |
                                                    +
                                                 XXXXXXXX
                                                XX      XX
                                                X        X
                                                XInternetX
                                                X        X
                                                XX      XX
                                                 XXXXXXX
```

Figure 3: Hybrid Wi-Fi to Packet Core integration model for Wi-Fi calling

4.1.3. Trusted Model

Enhancements introduced in 3GPP release 12 SaMOG specifications provides the ability to support multiple APN over Wi-Fi access making the support of Wi-Fi calling, and other applications specific APNs possible without the need for IPSec connectivity between the UE and the Packet core. This Architecture model is illustrated in figure 4 below

```
                                                             +-------+
                                                             |  IMS  |
                                                             | Core  |
            +-------------+              +-----------+       +-------+
            |             |              |           |           |
            | +--------+  | Flow mapped  |           |           |
            | |IMS APN |  | to VMAC-01   |           |       +-------+
            | |        +-----------------+           |       |IMS APN|
            | |Client  |  |              |           +-------+ P-GW  |
            | +--------+  |              |           |       +-------+
            |             |              |           |
            |             |              |Release 12 |
            |             |              |  TWAG     |
            |             |              |           |       +-------+
            |             |              |           |       |Default|
            |             |              |           +-------+ APN   |
            | +--------+  | Flow mapped  |           |       | P-GW  |
            | |Default |  | to VMAC-02   |           |       +-------+
            | | APN    +-----------------+           |           |
            | |Client  |  |              |           |           |
            | +--------+  |              |           |           |
            +-------------+              +-----------+           |
                                                               XXXXXX
                                                             XX     XXX
                                                            XX        XX
                                                           X           X
                                                          X            X
                                                          X Internet   X
                                                          X            X
                                                          X           XX
                                                           X         XX
                                                            XX    XXXX
                                                             XXXXXX
```

Figure 4: Trusted Wi-Fi to Packet Core integration model for Wi-Fi calling

4.1.4. Model Selection Criteria

Each of the Wi-Fi to Packet Core Architecture models described in the previous sections comes with its own pros and cons. And selection of a specific architecture model depends on several factors. Some of these factors, which can help determine the appropriate model, are listed below:

*Wi-Fi Access Network Ownership: There are several ownership models available when it comes to Wi-Fi to packet core integration. Wi-Fi Access network may be deployed by the MNO to leverage as another RAT to complement 3G and LTE. Alternatively the Mobile Network Operator may deploy a Managed Wi-Fi network for the Enterprise and SMB customers. The MNO managed Wi-Fi footprint is only portion of the overall Wi-Fi deployment. Third parties such as broadband service providers today own a significant portion of the Wi-Fi access network. For third party owned Wi-Fi access, the Mobile Network Operator may or may not have a direct roaming partnership with the Wi-Fi operator. The ownership model influences the choice of packet core integration architecture.

*Backhaul Network Ownership: From the context of this discussion here, the backhaul refers to the connectivity between WLAN Access network and the Packet core. It consists of a combination of wired access network of the hotspot, Broadband access last mile, Wi-Fi operator core network, Internet etc. These connectivity aspects will be deciding factor for the choice of Wi-Fi packet integration model. For example, Wi-Fi access network may be owned and or operated by the MNO, but if the backhaul involved a third party connection or Internet where MNO does not have control over security and QoS, an untrusted packet core integration may be the viable solution.

*Mobile Offload Requirements: Choice of the Wi-Fi to packet core integration model is not only influenced by voice offload but data offload as well. The untrusted Wi-Fi and the hybrid architectures do support a flexible offload model, allowing the Mobile Network Operator to choose which traffic to backhaul to the Mobile Packet Core to provide charging and added value services, while also leveraging local breakout capabilities on the device. Using the untrusted, and when applicable, the hybrid models allow the Mobile Network Operator to leverage their deployed network architecture for Wi-Fi calling. This makes both the hybrid and the untrusted Wi-Fi architectures valid options to consider depending on the Wi-Fi network ownership requirements.

*Device Capabilities: This greatly influences the choice of Wi-Fi to packet core integration. For example, a trusted approach with multiple PDN support requires the capability on the device to comply with 3GPP release 12 SaMOG enhancements, while the untrusted or hybrid model can leverage existing implementations and do provide a similar level of functionality.

*Support of Non-SIM devices: The MNO can provide value-added services, including voice services on Non-SIM devices The Untrusted Wi-Fi architecture is compatible with Non-SIM devices and provide the same capabilities to these devices as for the SIM devices.

*Network Readiness: This is another influencing factor for the choice of the trust model, as there are dependencies on the Packet Core network elements as well as Wi-Fi access network for the implementation of these models.

5. Subscriber Onboarding into Wi-Fi Access Network

Subscriber onboarding into a Wi-Fi access network is the process of getting connected to a WLAN access network and be able to offload mobile traffic successfully. In order to provide a seamless end user experience for Wi-Fi calling, the handset should be able to get connected to the WLAN with minimum or no user interaction. A seamless WLAN onboarding is critical for the smooth hand off of the voice call from LTE to Wi-Fi. There are several factors, which can influence the Wi-Fi onboarding experience. Proper choice of the available deployment options can ensure the subscriber onboarding experience is quite seamless.

5.1. Authentication and Identity Management

Before the UE device can successfully get associated with a WLAN access network it needs to get authenticated with the WLAN network. There are several types of user authentication options in use such as Web Portal based authentication, EAP-TTLS, EAP-TLS, EAP-SIM, EAP-AKA etc. Choice of the authentication mechanism depends up on the deployment preferences of the Wi-Fi operator. Web portal based authentication relies on an Open SSID configuration. Once the portal has successfully authenticated the UE device, the traffic is carried over the WLAN air interface without any encryption. EAP authentication mechanisms relies on secured SSIDs mandate the 802.11i based air encryption of the subscriber data in the WLAN access network.

In order to support Wi-Fi calling, one of the EAP based mechanisms will be preferred over the web portal based authentication. In the case of Web based authentication, the user needs to manually enter the username and password credentials or in some cases sign up for a service via Operator portal. But with any of the EAP methods, once the credentials have been established on the UE device, then authentication happens automatically without user intervention and greatly improves the onboarding experience.

If the Wi-Fi operator decides to use a secured SSID for subscriber authentication, choice of the EAP method depends up on the business model. A Standalone Wi-Fi operator may need to rely on non-SIM based EAP authentication mechanisms such as EAP-TTLS or EAP-TLS for their home subscribers. A Wi-Fi operator who has a roaming partnership with an MNO could allow the uSIM credentials of the MNO subscriber to be used for the access. In this case, the Wi-Fi operator will act as a proxy and authenticate the customer credentials with the MNO HSS.

Identity management deals with establishing subscriber identity and associated credentials on the UE device for WLAN onboarding. Identity management and authentication goes hand in hand. Option leverages the same set of identity and credentials (unified identity) for WLAN onboarding and packet core connectivity will simplify the identity management for Wi-Fi calling. However this requires that the WLAN access network is either owned by the MNO or by their roaming partner. With unified identity, typically uSIM credentials will be leveraged for both WLAN onboarding as well as packet core connectivity for SIM devices, and an EAP method used for Non-SIM devices.

5.2. Hotspot 2.0 for Seamless Onboarding

Ability for a handset to Seamlessly get connected to WLAN access network is one of the key factors which will influence the overall subscriber experience with Wi-Fi calling. Passpoint specifications defined by the Wi-Fi alliance under the Hotspot 2.0 program supports automatic discovery, selection and onboarding of Wi-Fi clients on to a compatible Wi-Fi access network. Figure-5 below is used to illustrate the hotspot 2.0 solution at a high level:

```
                              +----------------+
                              | Wi-Fi Operator |
                              |  AAA Server /  |
                              |  AAA Proxy     |
                              |                |
                              +----------------+
                                      |
                                      |
                                      |
                              +----------------+              XXXXX
           +-----------+      |                |            XXX   XXX
           |           |      |  +----------+  |           XX       XX
           |           |      |  |  ANQP    |  |          XX         XXX
           |    UE     |      |  | Server   |  |          X            XX
           |           |      |  |          |  |         XX   Roaming   X
           | +-------+ +------+  +----------+  +---------XXInterconnectXX
           | |  ANQP | |      |                |         XX            X
           | | Client| |      |                |          XX          XX
           | |       | |      |  +----------+  |           XX       XXX
           | +-------+ |      |  |  AP/AC   |  |            XX     XX
           +-----------+      |  |          |  |             XXX XXX
                              |  +----------+  |               X+X
                              +----------------+                |
                                      |                         |
                                      |                         |
                                      |                   +------------+
                                    XXXXX                 |   MNO      |
                                  XX    XXX               | AAA Server |
                                 X        XX              |            |
                               XX          XX             +------------+
                               X  External  X                   |
                              XX  Network   XX                  |
                              X    Access    X                  |
                              XX            XX            +------------+
                               XX          XX             |    HSS     |
                                XX       XXX              |            |
                                 XXXX  XXX                +------------+
                                    XXXX
```

Figure 5: Hotspot 2.0 with Service Provider Roaming

ANQP server is the component, which assists with the automatic discovery of WLAN network resources by the UE device. ANQP server is typically collocated on the Access Point (AP) or the Access Controller (AC). A Hotspot 2.0 compatible UE device will have a built in ANQP client. When a UE roams into the coverage area of a Hotspot 2.0 enabled network, it automatically learns about the network capability via Beacon or Probe Response. Then UE requests a set of network and service level information from the WLAN network. Based up on the info UE can decide which WLAN access is the most preferred and the type credentials it can use for getting connected.

5.2.1. Hotspot 2.0 Inter-Operator Roaming for Wi-Fi Calling

MNOs can enter into roaming partnership, which will allow Wi-Fi calling clients to automatically get connected to the WLAN access. This also allows the devices to leverage uSIM credentials or EAP credentials for Non-SIM devices for getting authenticated with the WLAN network. The Wi-Fi operator AAA will function as a proxy in this case and completes the authentication by interfacing with the MNO AAA Server and HSS, for EAP_SIM/EAP_AKA in the MNO packet core.

6. Wi-Fi calling deployment in restrictive networks

The use of IPSec to establish a connection to the ePDG, require that the access network allow IPSec tunnel establishment. But some networks won’t allow IPSec traffic either as a security policy or as a side-effect of only allowing "web traffic". In addition, many mainly corporate environments do deploy an HTTP proxy which will also prevent the establishment of an IPSec tunnel. Performing changes to these deployments may not always be possible or cost effective for the corporation or the public venues, especially in an "Untrusted Wi-Fi" model without the MNO involvement. In such situations, the mobile device can leverage the IPSec TCP encapsulation as described in draft-pauly-ipsecme-tcp-encaps-04 and in 3GPP TS 24302, which define the encapsulation of IPsec traffic in TCP. The Mobile device shall enable the TCP encapsulation only after failling to establish an IPSec connection to the ePDG. Figure 6 below shows the TCP encapsulation with the use for TLS to traverse a Proxy and reach the ePDG.

```
   +------------+                  +----------+                 +-------+
   | +--------+ |  TCP with HTTP  |          |  TCP with TLS   |       |
   | |  SWu   +===================+==========+=================+       |
   | | Client  ------------------------------------IPSEC-------        |
   | |        +===================+==========+=================+  ePDG |
   | |Proxy   | |   then TLS      |  Proxy   |                 |       |
   | |Config  | | through proxy   | Firewall |                 +-------+
   | +--------+ |                 +----------+                     |
   |    UE      |                                                  |
   +------------+                                               +-------+
                                                                |       |
                                                                | IMS   |
                                                                | PGW   |
                                                                +-------+
```

Figure 6: Use of TCP encapsulation for IPSec

When an HTTP proxy is deployed, the UE should connect to the eDPG through the proxy and then establish a TLS connection toward the ePDG. TLS is not used for securing the link, but to traverse the HTTP Proxy, and is configured with NULL-Cipher. This model allows Wi-Fi calling to operate even in restrictive networks.

7. RF Network Performance Optimization

Quality of the Wi-Fi calling experience would be as good or as bad as Radio network itself. Three network performance KPIs which impact the quality of voice are latency, jitter and packet drops. A healthy network is critical to ensure that these KPIs will meet the thresholds allowed to meet the acceptable voice quality. This section primarily talks about various performance optimization mechanisms available on the Wi-Fi Radio network.

7.1. Radio Resource Management

Radio Resource Management (RRM) aka Wi-Fi SON refers to the co-ordinated fine-tuning of the various RF network parameters among access points connected in a Wi-Fi network. It is very typical for Wi-Fi deployments from multiple operators to co-exist in the same hotspot. Scope RF fine tuning will be limited to the access points which are managed by the same operator in a specific hotspot. RRM fine-tuning will be typically performed by a centralized entity such as Access Controller. Some deployments which may not leverage AC such as Residential Gateways could leverage a cloud based RRM or SON Server. RRM controller continuously analyze the existing RF environment automatically adjust the power and channel configurations of access points to help mitigate issues such as co-channel interface and signal coverage. A proper implementation of RRM can greatly influence the RF performance and will have a positive impact on network KPIs that influence the Wi-Fi calling experience.

7.2. Wi-Fi Roaming Optimization

Roaming from the context of the discussion here refers to the hand of of a UE device from one Access Point to another Access Point in the same Extended Services Set (ESS) or mobility domain. Unlike cellular roaming between base stations, which is initiated by the network, in Wi-Fi the roaming is initiated by the UE device. A UE typically decides to disconnect from the current access point when some of the RF measurements such as RSSI, SNR etc. drops below certain threshold. There are other APs in the range with acceptable measurements the UE will start re-association process with one of the target APs. End user experience for a Wi-Fi call, which is active at the time of the hand off, will depend up on multiple factors. One critical factor is the time taken for the UE traffic to resume during the hand off. Also it is important that UE is able to make the optimum selection of the target AP from the list of available APs in the range. Discussed below are few IEEE 802.11 based mechanisms available to optimize the roaming.

7.2.1. Fast BSS Transition

IEEE 802.11r based fast BSS transition (FT) helps reduce the handoff time for a UE when it roams from one AP to another with in an ESS, which is enabled, with an EAP based authentication. Without FT, the UE will have to go through the full authentication process with the RADIUS server and device fresh set of encryption for 802.11i air encryption. When FT is enabled, the client will have an initial handshake with the target AP while still connected to the original AP. This handshake allows client and target APs to derive the encryption keys in advance to reduce the hand off time. Fast Transition can significantly improve the end user experience for the voice calls, which are active during a hand off.

7.2.2. 802.11k based Neighbor Reports

IEEE 802.11k enhancements allow a UE device to request from the current AP to which it is connected for a recommended list of neighboring APs for roaming. Up on receiving the client request, the AP responds with a list of neighbors on the same WLAN with the Wi-Fi channel numbers. Neighbor list is created by the AP based up on the Radio Resource Measurements and includes the best potential roaming targets for the UE. Neighbor list allows UE to reduce the scanning time when it is time to roam into a new AP in the same WLAN and there by improves the roaming performance. It is recommended to enable

802.11k along with Fast BSS transmission for optimum roaming performance.

7.2.3. 802.11v based Assisted Roaming and Load Balancing

Typical WLAN deployments will have APs with overlapping coverage areas. This is done on purpose to seamless handoff and also to address capacity requirements. Load distribution of UEs in the same coverage area may be helpful to proactively manage the bandwidth requirements and there by improve the subscriber experience. In the most rudimentary form, some of the load balancing solutions relies on the brute force method of ignoring the association requests from a UE by the APs with high load. Another more sophisticated mechanism is to leverage 802.11v based network assisted roaming. 802.11v allows unsolicited BSS transmission management messages from AP towards the client with a list of preferred APs to make roaming decisions. If the AP is experiencing high load, or bad connectivity from the client it may send an unsolicited BSS transmission management frame with the recommended list of APs to roam into. Depending up on the client implementation, it may or may not honor this info while making oaming decisions.

8. QoS Deployment Considerations for Wi-Fi Calling

This section covers the traffic prioritization mechanisms available in various segments of the overall traffic path of the Wi-Fi calling signaling and bearer sessions. Flexibility control of the QoS implementations will depend up on various factors such as ownership and management of the WLAN access network, Wi-Fi to packet core integration model etc.

8.1. Wi-Fi Access Network QoS

Traffic prioritization in the WLAN for Carrier Wi-Fi calls can be achieved by implementing Wi-Fi Multimedia (WMM). WMM consists of a subset of IEEE 802.11e enhancements for Wi-Fi. WMM defines four Access Categories, AC1, AC2, AC3 and AC4. AC1 is mapped against voice, AC2 is mapped against video, AC3 is mapped against best effort traffic and AC3 is mapped against Background traffic. Each of these Access Categories is mapped against one or more 802.11e User Priority (UP) values. UP has range from 0 to 7. Higher UP values typically gets more expedited over the air treatment EDCA mechanism for channel access defined in 802.11e is modified to make sure that traffic in higher UP queues get higher priority treatment. WMM can only leveraged if the client can do the right classification and Access points also support it.

8.2. End to End QoS

While QoS on the WLAN access network is critical, that by it may not be sufficient to maintain the subscriber quality of experience. It is important to enable QoS prioritization across all the network segments, which form part of the end-to-end voice path. Flexibility of the QoS implementation along the network segments will depend up on the trust models, which are discussed earlier. For example, if the transit path between WLAN network and Packet Core is include Internet, no QoS prioritization can be implemented over the Internet backhaul. How ever for deployment scenarios in which all network segments along the voice traffic path are managed either by the Mobile operator or their partners, then it makes much easier to implement end to end QoS. End to end QoS Classification for Wi-Fi calling is illustrated in figure 7 below.

```
                    Voice UP 6          Voice DSCP 46
                  Voice Sig. UP 4      Voice Sig. DSCP 24

                 +--------------+     +---------------+
                 | WMM or WMM+AC|     | DiffSrv QoS   |
                 |              |     |               |
                 v              v     v               v

             +----+           +--------+         +--------+
             |    |           |  WLAN  |         |        |
             |    |           |        |         |        |
             | UE |           | +----+ |         |  TWAG/ |
             |    +-----------+ | AP/| +---------+  ePDG  |  <------+
             |    |           | | AC | |         |        |         |
             |    |           | +----+ |         |        |         |
             +----+           +--------+         +--------+         |
                                                      |  Dedictated |
                                           Voice QCI 1|  and        |
                                           Sig. QCI 5 |  Default    |
                                                      |  Bearers    |
                                                      |             |
                                                 +--------+  <------+
                                                 |        |
                                                 |  P+GW  |
                                                 |        |  <------+
                                                 +--------+         |
                                                      |             |
                                                      |    DiffSrv  |
                                                      |      QoS    |
                                                      |             |
                                                 +--------+         |
                                                 |        |         |
                                                 |  IMS   |  <------+
                                                 |        |
                                                 +--------+
```

Figure 7: End-to-end QoS Reference Model

This QOS reference model assumes that, MNO or their roaming partners manage all the segments in the end-to-end path for voice signaling and voice bearer traffic. Model also assumes that transit path between WLAN and Packet core is private and secured and does not traverse Internet.

QoS reference model leverages WLAN access network leverages WMM that is described in the previous section, UP value of 6 is typically used for voice bearer traffic and UP value of 4 is used for voice signaling traffic. In order for voice to get the proper prioritization, WMM needs to be supported and enabled on both UE and the WLAN network.

In the transit IP network between WLAN and packet core, DSCP based QoS prioritization can be deployed if the connectivity is part of a managed transport. DSCP value of 46 is typically used for marking voice bearer and DSCP value of 24 is typically used for marking voice signaling. Proper traffic prioritization will depend up on whether DiffSrv QoS is enabled in the transit network.

Between P-GW and ePDG or TWAG, dedicated bearer with QCI value 1 will be established dynamically for voice calls. For signaling traffic a default bearer with QCI value of 5 will be used. These QCI values are mapped against specific QoS SLAs and allocation retention policies (ARP).

9. Wi-Fi Calling Client Considerations

Wi-Fi Calling client device functionality requirements depend on the on the models used for WLAN to packet core integration. At a minimum the clients should support IMS User Agent as defined in the 3GPP spec and be able to send and receive both IMS signaling and bearer traffic over a Wi-Fi access point. In addition, an SWu client that supports IPSec will can use ePDG-based packet core integration. This section talks about some of the client side implementation considerations for Wi-Fi calling.

9.1. Access Selection Criteria

The client device must select which RAT (cellular or Wi-Fi) it will use for communication to the cellular network. Commonly deployed access selection criteria is described below:

Device Local Policy Profile: In this case, the logic is defined by locally configured policy. Local policy may allow the end user to set prereferences. It is also possible for carriers to push these profiles to the device. Some MNOs may prefer cellular instead of Wi-Fi for voice service when both RAT technologies are available. Some other carriers may have Wi-Fi preferred approach for IMS APN when both RAT technologies are available. If Passpoint is enabled on the Wi-Fi access network, the client may take into account network loading conditions learned from the ANQP server to decide whether to offload IMS traffic into the Wi-Fi network.

9.2. Inter-RAT Handover

Inter-RAT handover refers to the handover of an active voice call without service disruption when the UE switches out from one RAT technology to another. Implementations must support handovers between Wi-Fi and LTE.

Handover between LTE and Wi-Fi is acheived by maintaining IP or IPv6 addresses between the LTE interface and the IPSec tunnel over Wi-Fi. If the IPSec tunnel is negotiated while a call is already in progress, the IKEv2 Configuration Request should specify the local address of the LTE interface in order to get assigned the same address on the IPSec tunnel. Similarly, handover from an IPSec tunnel over Wi-Fi to LTE requires the LTE interface to be brought up with the same address as the tunnel. Maintaining the address allows the client to not interrupt TCP or UDP connections that are using the local address for communication. In a system that uses POSIX sockets, for example, the handover must be done in such a way that the sockets do not need to be closed and re-opened.

9.3. MTU Considerations

When handing over between LTE and IPSec tunnels over Wi-Fi, the client device should be aware of the Maximum Transmission Unit (MTU) of each interface. It is possible that the effective MTU for the IPSec tunnel (which can be calculated as the MTU of the Wi-Fi interface minus the overhead for ESP encryption) is notably smaller than the effective MTU of the LTE interface. For UDP flows, they should avoid sending large datagrams that could get fragmented when handing over between RATs. For TCP flows, the Maximum Segment Size based on the MTU SHOULD be re-calculated upon handover.

9.4. Congestion Management

Radio Network Performance management and QoS considerations described earlier can significantly contribute to the overall QoE for Wi-Fi calling. A client driven congestion management mechanism can positively augment the overall experience. The idea is to dynamically change the bandwidth requirements for the call based up on the network congestion conditions. Network resource requirements (bandwidth, packets per second etc.) per call are directly proportional to the type of codec and the packetization rate. Sometimes it may be desirable to switch out to a lower audio codec to keep the drop, delay and jitter characteristics under acceptable levels during periods of network congestion. Explicit Congestion Notification for RTP over UDP defined in RFC 6679 can be used to inform network congestion to the end clients. But this requires the network elements to mark the ECN bits on the IP header of the packet when congestion conditions are encountered.

9.5. NAT Traversal

Since NATs are very commonly deployed primarily due to the shortage of IPv4 address space, a client side implementation should support NAT traversal for Wi-Fi calling. IPSec implementation on the client side should support the detection of NAT gateways as defined in RFC 7296 specification. If a NAT gateway is detected, client should send all subsequent IPSec traffic from port 4500. If NAT is detected ESP packets must be UDP encapsulation using port 4500. If NAT devices are not detected, SWu may use pure ESP encapsulation without UDP. It is important to understand the implications on firewall rules with and without NAT so that the Wi-Fi calling does not get blocked by the firewall. Many deployments may allow ESP with UDP encapsulation by default but may block ESP only tunnels.

10. Acknowledgements

Authors would like to acknowledge the inputs and advice provided by Eduardo Abrantes and Ajoy Singh.

11. Informative References

[IR51] "IMS Profile for Voice, Video and SMS over untrusted Wi-Fi access Version 5.0", 2017.

[IR92] "IMS Profile for Voice and SMS Version 10.0", 2016.

[RFC4066] Liebsch, M., Ed., Singh, A., Ed., Chaskar, H., Funato, D., and E. Shim, "Candidate Access Router Discovery (CARD)", RFC 4066, DOI 10.17487/RFC4066, July 2005, <http://www.rfc-editor.org/info/rfc4066>.

[RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, DOI 10.17487/RFC4187, January 2006, <http://www.rfc-editor.org/info/rfc4187>.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <http://www.rfc-editor.org/info/rfc4555>.

[RFC4881] El Malki, K., Ed., "Low-Latency Handoffs in Mobile IPv4", RFC 4881, DOI 10.17487/RFC4881, June 2007, <http://www.rfc-editor.org/info/rfc4881>.

[RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <http://www.rfc-editor.org/info/rfc5213>.

[RFC5568] Koodli, R., Ed., "Mobile IPv6 Fast Handovers", RFC 5568, DOI 10.17487/RFC5568, July 2009, <http://www.rfc-editor.org/info/rfc5568>.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <http://www.rfc-editor.org/info/rfc5944>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[TS22228] "Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS); Stage 1", 2010.

[TS23402] "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Architecture Enhancements for non-3GPP Accesses.", 2009.

[TS23852] "Study on S2a Mobility based on GPRS Tunneling Protocol (GTP) and Wireless Local Area Network (WLAN) access to the Enhanced Packet Core (EPC) network (SaMOG); Stage 2", 2011.

[TS29273] "Evolved Packet System (EPS); 3GPP EPS AAA interfaces", 2011.

Authors' Addresses

Byju Pularikkal
Cisco Systems
170 West Tasman Drive
San Jose
United States

Email: byjupg@cisco.com

Tommy Pauly
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
US

Email: tpauly@apple.com

Mark Grayson
Cisco Systems
10 New Square Park
Feltham
United Kingdom

Email: mgrayson@cisco.com

Sri Gundavelli
Cisco Systems
170 West Tasman Drive
San Jose
United States

Email: sgundave@cisco.com

Samy Touati
Ericsson
300 Holger Way
San Jose, California 95134
US

Email: samy.touati@ericsson.com

Operations and Management Area Work Group
Internet-Draft
Intended status: Informational
Expires: September 27, 2017

S. Rich
Cisco Systems

T. Dahm
Google

27 March 2017

MUD Lifecyle: A Manufacturer’s Perspective
draft-srich-opsawg-mud-manu-lifecycle-01.txt

Abstract

Manufacturer Usage Descriptions, or MUDs, allow a manufacturer to cheaply and simply describe to the network the accesses required by an IoT device without adding any extra cost or software to the devices themselves. By doing so, the network infrastructure devices can apply access policies automatically which increase the overall security of the entire network, not just for the IoT devices themselves. This document describes the lifecycle of Manufacturer Usage Descriptions (MUDs) by describing detailed MUD scenarios from the perspective of device manufacturers.

Status of This Memo

1. Introduction

The addition of IoT devices to a network expands the attack surface of that network. Even if a device does not have exploitable vulnerabilities (in the sense of an attacker injecting and running malware on it), it may be susceptible to denial-of-service (DoS) attacks and thus could have its functionality impaired by attackers. Recent events have shown just how real, and not just theoretical, such attacks can be.

A detailed summary of the current state of understanding of the Mirai botnet's use of IoT devices can be found in [MIRAI]. It is estimated that around 100,000 IoT devices generated more than a terabit per second of DDoS traffic.

Also consider the Sony Cameras IP Security article [SONYCAMS] which describes a vulnerability in many camera models which could be exploited to launch attacks like those seen in the massive DDoS attack on DynDNS in [DynDNS]. As both of these incidents show, more network-accessible devices which can connect to arbitrary external addresses can, if those devices permit too much access or if they have vulnerabilities which allow arbitrary code execution, be used by attackers to amplify attacks and to do so by using origin addresses spanning broad ranges of networks.

Concerns about the negative possibilities of attacks related to IoT devices is also discussed in [MITTECH] that also discusses some of the regulatory and government angles in play. In a recent move described in [USGSUIT], the U.S. Federal Government has taken the step of suing D-Link, accusing it of ‘‘poor security practices’’ for some of its IoT devices.

MUD provides a light-weight model of achieving very effective baseline security for IoT devices by simply allowing a network to automatically configure the required network access for IoT devices so that they can perform their intended functions without granting them gratuitous, unrestricted network privilege.

2. MUD High-level Introduction

Manufacturer Usage Descriptions (MUDs) provide advice to end networks on how to treat specific classes of devices. The MUD architecture is explained in [LEAR2017], but we will describe it briefly here and also discuss details where necessary to understand this document. At its most basic, MUD is a system by which the IoT device itself tells the network exactly how to retrieve its network access requirements (in a ‘‘MUD File’’, which is the term used in the MUD specification to refer to the file which contains the description of an IoT device’s network access requirements), and network infrastructure can fetch and act upon this information. The MUD File itself is a static text file which the network infrastructure element responsible for it can retrieve from the manufacturer or from whomever the manufacturer delegates the responsibility to. The MUD file may be cached, so when served, the MUD file should be returned with a ‘‘max-age’’ value which lets the requestor know how long it can cache it.

To add MUD support to an IoT device is a very minimal change: add the URL for the MUD File as the ‘‘MUD URI’’ to whatever dynamic network registration protocol which is currently being used by the device (e.g. DHCP, etc.). It is so simple that the device manufacturer can statically compile the URI into the firmware of the device. The essential point is that MUD does not force a large behavioral change on the IoT device itself, and the serving up of the MUD file during the lifetime of the devices is similarly relatively low-impact. The bulk of the complexity of MUD is concentrated within the network elements which perform operations to retrieve the MUD files, possibly cache them, and then configure the network in response, but even there, the network elements effected mostly already perform all of these actions, albeit not automatically in most cases.

For this description, one can consider three general classes of actors in the MUD ecosystem:

- o Device manufacturers

- o Networking equipment manufacturers

o Network operators

Note that end users are not mentioned here, as their involvement in MUD is minimal at best (and likely only present in the simplest of deployments). Note also that ‘‘Device manufacturers’’ are described with the assumption that they will both include MUD URIs within their devices as well as service MUD URL requests (via a cloud service or via their own web infrastructures). It is possible that a manufacturer will delegate the MUD URL retrieval function to a third party. The question of who actually services network requests for the MUD URL is an administrative one and does not affect the MUD architecture. It does give device manufactures more flexibility, though, in managing their investment into the MUD ecosystem.

This document will describe the MUD ‘‘lifecycle’’ from the standpoint of manufacturers, but it is also intended to be informative to persons interested in standardization, installation, or other areas where MUD may be in play. Where appropriate, suggestions of best practices will be given if there are no specific hard requirements.

3. Terminology

Before going into descriptions how MUD works, we will list terms used within the MUD ecosystem:

MUD
: Manufacturer Usage Description

MUD file
: a file containing YANG-based JSON that describes a recommended behavior

MUD file server
: an HTTPS server that hosts a MUD file

MUD controller
: the system that requests and receives the MUD file from the MUD server. After it has processed a MUD file it may direct changes to relevant network elements

URL
: Universal Resource Locator

URI
: Universal Resource Identifier. The difference between a ‘‘URI’’ and a ‘‘URL’’ is that a URI is intended to be used as an identifier in a general sense, whereas a URL is a specific use case of a URI that is used to access something at a particular network location

MUD URI
: a URI that an IoT device carries and which will be issued during operations such as DHCP requests which can be used as a URL to retrieve a MUD file

MUD URL
: the MUD URI being used as a URL

IEEE 802.1AR
: A IEEE specification for a certification-based approach for communicating device characteristics

YANG
: A data modeling language for the definition of data sent over the NETCONF network configuration protocol [RFC6020]

NETCONF
: Network Configuration Protocol [RFC6241]

JSON
: Javascript Object Notation, a human- as well as machine-readable file format containing textual representations of ‘‘objects’’ such as strings of characters, numbers, boolean values, and lists and dictionaries of such objects and collections of objects

Many of these terms are in common usage with the IETF or other network standards bodies and are thus used for consistency. More information about terms like ‘‘URL’’, ‘‘URI’’, ‘‘YANG’’, and ‘‘NETCONF’’ can be found in the standards and references published by the IETF and others. The value in distinguishing ‘‘URI’’ and ‘‘URL’’ will hopefully become more apparent when MUD file caching is discussed (during which time, already-retrieved MUD files will be used if the URI lookup returns a match). The actual text of a ‘‘MUD URI’’ and a ‘‘MUD URL’’ will generally be identical; the distinction lies in the use of it by various elements (IoT devices, network devices, and web services).

4. MUD Operation

```
+--------------------------+ +-------------------------------------------+
| [Manufacturer]           | | [Customer]                                |
|            -------       | |                                           |
|     +----->(       )  6 | | +------------+  7  +----------------+      |
|     |      |-------|<--------|    MUD     |---->| Network Policy |      |
|     | 2:MUD|       |-------->| Controller |     |   Management   |      |
|     | File (       )       | | +------------+     +----------------+    |
|     |       -------        | |           ^        |                     |
|     |                      | |   5:MUD   |        | 8                   |
|  (1:Create Device)         | |     URI   |        v                     |
|          |                 | |         +-------------+                 |
|          |                 | |         |  (Switch)   |                 |
+----------|-----------------+ |         +-------------+                 |
           v                   |          |  |  |  |  |                  |
+--------------------+ 3:Buy   | 4:Deploy |  |  |  |  |                  |
| Distribution    ----------->|          X  X  X  X  X  ...             |
| Channels           |         |                                         |
+--------------------+         +-----------------------------------------+
```

    6: MUD URI used as URL to request MUD File
    7: MUD Controller informs network policy engine about ACLs
    8: Network policy applied as close to IoT device as possible

Figure 1: MUD-related network information flow

A full description of MUD is given in [LEAR2017]. In short, when a device such as an IP-enabled lightbulb is connected to the network and given power, that device will perform some action to acquire a network identity, including an IP address, such as by making a DHCP request. If that request has a MUD URI in it, equipment in the network (not necessarily the DHCP server) can use that URI to retrieve the device’s MUD file from the MUD file server. Some other networking component (the switch to which the bulb in connected, for example) can then act on the contents of the retrieved MUD file and apply the appropriate configurations to allow the device to function normally while restricting where it can connect.

A MUD file’s contents will mostly contain descriptions of which protocols are required by the device and over what port or ports.

From the perspective of a manufacturer, the essential elements to note are the following:

1. On the device itself, the only change required to add MUD compliance/functionality is to add a field populated with a URI to whatever network access protocol is already being used (i.e., DHCP, IPv6 AD, etc.). This will be a static text string which will probably remain constant throughout the life of the product and which is identical for every instance of a product run (i.e., there is no per-serial-number version of the MUD URI)

2. The MUD file which is to be returned via an HTTPS server can be a static file and can be reused for devices which have the same network access requirements. The service which returns the MUD file will not be responsible for any security policy enforcement, as that is the job of the network which contains the devices themselves

3. MUD files are fairly short (on the order of tens of lines of text) and are thus trivial to serve either directly and are amenable to caching

4. The act of retrieving the MUD file and of acting on it is entirely up to the network infrastructure and not a responsibility of the IoT devices themselves. MUD does not impose any behavioral requirements on the IoT devices themselves other than that they must send the MUD URI during network access configuration, as mentioned earlier

How does MUD work in practice? Figure 1 shows a representation of the high-level MUD information flow. This document deals almost exclusively with elements in the upper left of that figure. Specifically, it describes what a manufacturer should do to put a MUD file into a device and what is required for a manufacturer (or a designee of the manufacturer) to answer requests for MUD files from network operators whose networks provide connectivity for such devices.

5. Device Manufacturer Considerations

The device manufacturers have the most insight into what resources the devices will need once they are installed in a network. They are thus best-suited to author the network profiles which will be required by the devices that they make for correct operation. Conversely, each manufacturer cannot know what each network’s other requirements happen to be. As a result, the manufactures should provide configuration requirements for their devices which network operators can apply in a way best suited for their networks. The network operator can optimize operations through caching, LAN segregation, etc., and can use the MUD information to further secure the network.

If a manufacturer makes many devices which have similar network access requirements, that manufacturer may want to leverage common profiles. They should do so only when the profiles are truly close enough to be treated as the same.

Device manufacturers have three responsibilities under MUD:

- They must author a MUD profile which describes a device’s requirements for network access

- They must encode a MUD URI into the device such that when the device performs DHCP or similar

- The MUD File must be hosted on a publicly-available web server

Since the MUD profiles can be static files, there is very little overhead required to serve these profiles. Due to their static nature, they are inherently cacheable.

Similarly, since the URI can be essentially static (the actual device configurations are easily updatable since they are contained in the MUD file, not the URI), the manufacturer can assign a name space and begin encoding the URIs into the devices relatively early in the manufacturing process, including before the MUD specification is finalized. An important point is that manufacturers should adopt and follow a nomenclature that insures that they can sufficiently distinguish classes or families of devices with different requirements and assign them different URIs. From a security standpoint, it is better to have several URIs with more granular security profiles than it is to have a very few URIs with "catch-all" (and thus more open) security profiles. This ensures that a customer using a single family of devices will have the most closed network configuration possible.

If the device manufacturer decides to update the profile, then it may do so at any time, independently of updates to the firmware on the devices themselves. If it is expected that a profile may change frequently (say, for a new class of devices which aren’t fully understood yet), then the MUD profile for said device should be served with a fairly short max-age (as compared to a device with a well-established network access profile).

6. High-level MUD Lifecycle

The following lifecycle description is described considering a single device. As additional devices are added to a portfolio, the same steps are taken for each one where necessary. Each step can be isolated or coordinated with other device instances where convenient. There is little coupling inherent in the way that the various phases of MUD deployment operates to impose strict requirements in this area.

1. Based on a device’s function, a MUD profile is either:

   - Chosen from a library of existing profiles for similar devices

   - Written anew to describe this device’s network requirements

2. If the profile is pre-existing, the a choice is made if this device will receive a new URI or if it should be classed as identical to existing devices and use the same URI

3. The chosen URI is assigned to the device so that when the device performs network initialization, the URI is included in the request (i.e., DHCP, ANIMA, etc.)

4. In parallel or in advance (but prior to first customer shipment), the device manufacturer should allocate in an appropriate namespace and place the MUD profiles for when the URI is used as a URL.

5. The MUD profile should be made available to customers until such a time that the device is unsupported. While it is outside the scope of this document, The manufacturer should support MUD profile retrieval for each device for at least as long as the manufacturer supports the devices themselves.

6. If the profile is found to contain an error, the manufacturer should update the profile. Devices which are already deployed will continue to use the original URI (unless a firmware updates changes it), so the original profile should be corrected

7. If a device manufacturer chooses to update a MUD-enabled device’s firmware, the manufacturer may update the MUD URI to a new one. The manufacturer should change the URI if the network access requirements of the new firmware are sufficiently different from those of the original firmware version.

7. MUD URI

The MUD URI is a very visible and important part of MUD that is best done correctly from the start, for once it is embedded in an IoT device, changing it for the fielded devices will be, at best, inconvenient. Choosing a scheme for organizing the ''name space'' for the portion of the URI which is controlled by the device manufacturer may have knock-on effects such as the URL GET request routing behavior that must be supported during MUD file retrieval.

The format of the URI is:

https://authority/.well-known/mud/mud-rev/model

where ''mud-rev'' is currently the literal string ''v1'', and may be suffixed with '' ?extras ''. Referencing [RFC3986], the authority element is described by the ''authority'' type, the model element by the ''segment'' type, and extras by the ''query'' type. This gives considerable flexibility to manufacturers to structure their various namespaces to handle a huge variety of device types. However, this document will restrict itself to describing a very simple URI encoding scheme.

In the following, we will use ''example.com'' as the authority element. By far, the simplest method of assigning MUD URIs to devices is to assign each distinct model number a URI of the form

https://example.com/.well-known/mud/v1/model

where the ''model'' element is literally the model number of the device. If a manufacturer has a model number collision problem (possibly because of acquisitions of other companies, for example), a simple scheme of a prefix or a suffix, set off with a hyphen or similar, will suffice to disambiguate them. Since the MUD files are relatively small, there is likely little value in conjuring schemes to save disk space with complicated naming conventions or structure.

8. MUD File Serving: Operations, Lifetypes, and Transfer

The previous section discussed how one might design the URI namespace for MUD files. Another very important consideration is the total lifecycle of the serving of MUD files via the internet for an appropriate length of time and what to do if one wants to transfer the responsibility of serving MUD files to some other entity. This section will describe several scenarios and suggest options for the transfer of responsibility of MUD files to other providers. There is no single set policy for these various activities, and organizations are free to decide how and when these transfers occur. There are technical considerations that must be dealt with, but this is not unlike outsourcing subsections of one’s web site to payment partners or other specialists if so desired.

The single largest factor in thinking about serving MUD files throughout their lifetimes is the relative ‘‘permanence’’ of the URI itself (since, for some types of devices, at least, the buried-in URI will be essentially indelible). Even if a device has a more fungible MUD URI (say, because it is easily and frequently updated), it is still wise to consider the case when a device’s MUD URI cannot be easily updated since this represents the most problematic case. Networks containing the MUD-enabled devices will make network requests to retrieve the MUD files. The MUD URIs are, quite literally, the URLs of the MUD files. There, network infrastructure devices from potentially anywhere on the internet will try to retrieve these MUD files. The volume of requests will be simple to handle (given that MUD files are static and small and that MUD servers in the network will be able to cache them and avoid redundant retrievals).

A very simple and direct way to manage MUD files and make the possible future delegation of MUD file serving to a 3^{rd}-party is to assign a URI DNS ‘‘namespace’’ for your company’s MUD files. For example, using the fictional company ‘‘Acme Lightbulb and Sensor’’ and its web presence at ‘‘https://acmels.com’’, the DNS namespace for MUD files could be

mud.acmels.com

which can serve as the authority section of the MUD URI. If Acme wants to serve the MUD files themselves, then they can provision an HTTPS service that serves that address and return the requested MUD files, or they can create a CNAME to point to the actual entity who will answer the requests.

9. Security Considerations

The bulk of this document describes the use of MUD to increase the security of a network. However, it is possible to compromise the effectiveness of MUD by attacking its behavior directly. This section discusses the known attacks and describes possible mitigations (all from the manufacturer’s perspective). This section also attempts to clarify the limits to which MUD is expected to perform in terms of increasing security.

The first and most obvious attack scenario is that a malicious or compromised device can issue a MUD URI which allows that device to communicate too permissively, either by having the URI refer to an unintended file or by simply putting too permissive a set of rules in the otherwise-legitimate MUD File. A manufacturer SHOULD employ secure development best practices to take reasonable steps to insure that their devices behave correctly at least up to the point that they are shipped and that their web services follow all BCPs.

Other attacks are not manufacturer-specific and will not be covered in this document. They will instead be discussed in TBD which focuses on the network operator’s perspective of MUD.

10. IANA Considerations

This document has no actions for IANA.

11. Normative References

[LEAR2017]
Lear, E., "Manufacturer Usage Description Specification", draft-ietf-opsawg-mud-03, January 05, 2017

[WEIS2017]
Weis, B., "RADIUS Extensions for Manufacturer Usage Description", draft-weis-radext-mud-00, October 25, 2016

[RFC6020]
Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", IETF RFC 6020, 2010

[RFC6241]
Enns, R., Bjorklund, M., Schoenwaelder, J. and Bierman, A., "Network Configuration Protocol (NETCONF)", IETF RFC 6241, 2011

[RFC3986]
Berners-Less, T., Fielding, R., Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", IETF RFC 3986, 2005

12. Informative References

[RFC2882]
Mitton, D., "Network Access Servers Requirements: Extended RADIUS

Practices", RFC2882, July 2000

[MIRAI]
https://www.flashpoint-intel.com/action-analysis-mirai-botnet-attacks-dyn/

[SONYCAMS]
http://www.pcworld.com/article/3147311/security/backdoor-accounts-found-in-80-sony-ip-security-camera-models.html

[DynDNS]
http://www.pcworld.com/article/3134056/hacking/an-iot-botnet-is-partly-behind-fridays-massive-ddos-attack.html

[MITTECH]
https://www.technologyreview.com/s/603015/security-experts-warn-congress-that-the-internet-of-things-could-kill-people/

[USGSUIT]
https://www.cnet.com/news/d-link-lawsuit-ftc-security-hackers/

Authors’ Addresses

Steven Rich
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134

Email: srich@cisco.com

Thorsten Dahm
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043

Email: thorstendlux@google.com

Operations and Management Area Work Group S. Rich
Internet-Draft Cisco Systems
Intended status: Informational
Expires: March 27, 2017 T. Dahm
Google
27 September 2017

MUD Lifecyle: A Network Operator’s Perspective
draft-srich-opsawg-mud-net-lifecycle-01.txt

Abstract

This memo describes the lifecycle of MUD as seen from the perspective of a network operator. It is informational and intended to help provide perspective around the operation of a network which connects MUD-supporting devices and uses MUD-supporting network infrastructure. All phases of network operation that involves or affects MUD will be described. Considerations specific to device manufacturers will be described elsewhere. Considerations relevant to network equipment manufacturers and networking software authors will be described where appropriate where MUD behavior is affected.

Status of This Memo

1. MUD Introduction

Network architects and operators have the goal of designing and operating networks so that they are reliable, secure, and operate correctly. Making them do so requires that the network permit traffic which is intended to be allowed on the network while rejecting or blocking traffic which is not. Both goals are met with a combination of policies and configurations which promote efficient routing of packets for certain classes of traffic and which rate limit or even block (possibly by black-holing) other classes of unwanted or lower-priority traffic.

A common assumption is that devices on the inside of the network can have relatively unrestricted access to other parts of the network and to the local network segment. This is reasonable for devices which themselves have certain configurations which will naturally govern which network access they require. For example, a printer will usually be configured to accept connections from hosts which wish to print to it. The printer itself may not tend to initiate outbound connections and thus does not require a complex set of custom ACLs. If the printer needs external connectivity, the usual scenario is to allow the printer to make outbound connections while still preventing inbound connections using a stateful firewall rule or similar. However, there are often no rules preventing the printer from making arbitrary connections within network delineated by the firewall.

Other devices such as general-purpose end-user hosts (PCs, Mac, etc.) might need unrestricted access, at least in the outbound direction, because, contrary to the printer example, end-user hosts are generally expected to make outbound connections to an unpredictable number of hosts. Even if outbound restrictions to certain ports (such as 80, 443, 22, 25, etc.) are enforced, the destination address may be unrestricted. As stated above, restrictions from internal hosts to internal addresses may be even more lax.

Enter into this situation IoT devices which may be introduced to the network in the thousands and which may have unspecified or unclear requirements for network access. For example, IoT light bulbs may need to talk to DNS, NTP, LLDP, DHCP, and a controller on the local network and nothing else. An IoT thermostat may need to talk to DNS, NTP, LLDP, DHCP, and its cloud-based controller, but nothing else. For both of these cases, while their specific requirements vary, knowing each one’s requirements would allow a tight set of ACLs to be imposed, all the way to the port level, to limit what connectivity is afforded to each individual instance.

Recent examples of IoT-based malware campaigns will not be repeated here and the benefits of providing such security will no doubt be obvious to network operators. What has not been available before MUD is an ability to automatically retrieve configuration policy and then automatically apply it for each device. This document will describe the ‘‘lifecycle’’ of MUD from the perspective of a network operator. The details of the protocol and contents of the MUD file itself are described in [LEAR2017], and familiarity with it is assumed for this document.

2. Terminology

This document will use some terms and abbreviations which will be listed and described in this section.

MPD
"MUD-Protected Device" - While this is a possibly tedious use of a three-letter acronym, repeated use of "MUD-protected device" or similar is equally tedious

AAA Server
"Authentication, Authorization, and Accounting Server" - A network service which processes AAA requests

ACL
"Access Control List" - In the context of this document, an ACL will refer specifically to those which are specified in a MUD file and which get applied at some point in the network to enforce the security policy needed by a device. These ACLs may be configured down the port into which the device the is plugged, and they may be applied "dynamically" in the sense that they appear in response

to the MUD request as opposed to a static configuration. They will not be dynamic in the sense that they change frequently. The actual implementation by any particular vendor is left up to that vendor and thus may differ from the examples given in this document.

3. MUD Lifecycle Description for Network Operators

The totality of what network operators must do to build, operate, and maintain networks will not be described in exhaustive detail in this document. Instead, we will describe what additional or different things are necessary or recommended when establishing MUD support within the network. Some of the steps discussed will presuppose that networking equipment vendors will have added MUD support to their products.

The following high-level tasks are required to support the automatic network configuration aspects of MUD devices on the network:

1. Network Segmentation Considerations and Design

2. Install and/or enable a MUD Policy Server

3. Configure network devices so that they will receive and enforce ACLs generated by the MUD Policy Server

4. Test and verify functionality by confirming that MUD files are retrieved and ACLs are applied to the appropriate ports and that those ACLs are removed when the port goes down

The MUD Policy Server may support caching retrieved MUD files. If it does, then the operator may choose to enable, tune, test, and monitor this functionality as well. Details about caching MUD files as well as each task above will be covered later in this document.

The network equipment to which MPDs connect must be capable of accepting and enabling dynamic ACLs which can preferrably be scoped to a port. While it is conceivable that the ACLs be combined and applied at a point in network that is multiple hops away from the switch to which the MPD connects, the tightest security controls are possible when enforcement can happen directly on the port. This eliminates the possibility that a MPD can talk to other devices on the same switch unless explicitly permitted. The remainder of this document will only discuss the case of using ACLs.

3.1. Network Segmentation Considerations and Design

A well-designed network is one which includes the use of segmentation which keeps different parts of the network isolated from each other to the optimimum degree. For example, groups of machines which need to communicate frequently and at high speed most likely should be on the same LAN. Different groups of machines which rarely communicate together can be separated into different routed networks, and depending upon security requirements, may even be guarded by ACLs or other mechanisms.

Different network segments may be designed with different expectations of security. Inner-bastion networks may contain sensitive systems which are isolated from all but the most trusted systems. Segments which allow guest users or devices which are less trusted may be relegated to segments which have also been protected with ACLs, but the focus can be on limiting what the devices in the segment can access rather than worrying about what external devices can access inside the segment itself.

The goal of MUD is to enable the near-automatic management of device segmentation for the class of devices which have MUD support. To be maximally effective, though, the network designer should take advantage of pre-defining segments into which MUD-capable devices can be grouped by function and by required access. An optimal middle ground (for a large network with many types of MUD-enabled devices) would comprise some device-class-specific segments, some vendor-specific segments, the essential set of network segments (required regardless of MUD for the normal operation), and perhaps a ‘‘default network’’ into which untrusted devices are placed which get no internal network access and severely limited internet access.

Ideally, with full MUD support in devices deployed in a network, there would be no need for the so-called ‘‘default network’’ segment (except perhaps as a ‘‘guest’’ network) since MUD profiles would result in a properly-segmented and protected devices. Until MUD is ubiquitously supported, though, it is wise to consider the option.

To make these ideas more clear, an example network will be described (at a high level) with various segments defined. The use of each segment by MUD will then be described. These are segments within a larger network which will not be described to avoid cluttering the diagram.

Segment Name	Segment Description	MUD
SecDB	Recorders, IDs	N
Readers	Badge Scanners	Y
Cameras	Security Cameras	Y
Other	Other IoT devices	Y
NetMgmt	Network Management	N

There are five segments. Two of them will have no MUD-enabled devices in them, whereas the other three will. Of those three, one is a non-classed MUD network (i.e., one in which MUD-enabled devices which do not belong to specifically-configured classes will be placed). The connectivity of the network looks like:

```
+--------------------+
| Network Management |-----------|
+---------------------+          |
  |                              v
  |                            +--------------+
  |                            | SecDB        |
  |  +---------+               | +----------+ |
  +->| Readers |---------------->|Controller| |
  |  +---------+               | +----^-----+ |
  |                            +-------|-------+
  |     +---------+                   |
  +---->| Cameras |-------------------+
  |     +---------+
  |
  |       +-------+
  +------>| Other |
          +-------+
```

The SecDB segment will contain senstive systems as well as a controller to which some of the other devices will need to communicate. The Readers will be a segment in which all badge/card readers will be grouped. The Cameras segment will contain all of the security cameras. Finally, all other MUD-enabled devices will be placed in the Other segment. Devices placed into any of these segments as a result of MUD will still have applicable ACLs applied. In addition, any static access control restrictions given to each segment will be enforced per the network designers’ intentions.

How do the cameras get into the Cameras segment, and how do the card readers get into the Readers segment? The specifics will depend on the MUD Controller implemenation and the network devices used, but the gist is that the network administrator defines policies which map a MUD file’s ‘‘manufacturer’’ and ‘‘model’’ to the appropriate network segment assignment policy. If no specific mapping is available for a device, then the MUD-enabled device will be placed into a default segment per the operation of the MUD Controller in use.

Another consideration is what to do with devices which have no MUD profile at all. This was the case for all device before MUD was defined and may continue to be the case for certain classes of devices. The solution again lies with the definition of network policies. It is up to the network designer to choose which segment or segments devices which have no MUD support are placed by default. Theoretically, the placement could be influenced by the MAC address, the port into which the device is plugged, etc.

The bottom line is that MUD is not responsible for fully describing the network configuration policy. It is very helpful to automatically limit the access that MUD-enabled devices are afforded to only what they need, but the network operator must insure that the network design is complete.

3.2. Installing and/or Enabling a MUD Controller

MUD Policy Servers can conceivably take on many forms, including stand-alone appliances, software modules installed on a switch or a router, a software package installed and integrated with a DHCP server, etc. The key requirements for MUD Policy Servers are:

1. Able to "see" a MUD URI

2. Able to retrieve a MUD file

For a MUD Policy Server to ‘‘see a MUD URI’’, it must either be able to see the DHCP or equivalent requests from MPDs directly or it must be otherwise connected to the service which does get to see these types of requests. For example the MUD Policy Server could be implemented as a plugin to a RADIUS server which is receiving requests from a switch which is handling DHCP requests by generating corresponding RADIUS AAA requests.

For a MUD Policy Server to be able to retrieve a MUD file, it must have network access permissive enough to retrieve files which are served from arbitrary locations on the internet.

Finally, to have any useful effect, the MUD Policy Server must be able to, having parsed a MUD file, generate ACLs which are to be applied to the appropriate port of the appropriate network device (i.e., a dynamic configuration must be generated and applied which reflects the MUD policy). The specifics of how the generated ACLs get back to the NAS and get applied to the proper port will depend on the design of the network.

At the time of this document’s preparation, MUD is still a new protocol and is under development. Therefore, descriptions of how it is integrated will be subject to adjustment according to the progression of actual implementations.

3.3. Network Device Configuration

There are two distinct "network configuration" concepts involved in the deployment of MUD:

1. Configuration of the network infrastructure such that the MUD controller is "in the loop" and able to issue configurations for devices as they appear on the network

2. The per-device dynamic configuration that is generated through the behavior of MUD itself

This document discusses both concepts where applicable. To avoid confusion, when a reference is made to "configuring a device" or similar, we will be referring to setting up the network infrastructure to include the MUD Policy Server into operations. The actions of the MUD infrastructure and network infrastructure to effect changes to network configurations persuant to MUD-advised policies will be referred to as "applying device policy" or (when it is more clear to do so) "applying the dynamic device configuration". The key word in the latter is dynamic and may be used when describing the specific steps being taken by the devices to apply the policies.

As previously mentioned, the ideal point for the application of MUD-based access restrictions is the port into which a device is directly plugged since this results in the most finely-grained application of access control and insures that devices are not able to talk even to neighbors on the same shared media without MUD authorization. For this to happen, the switches which connect to MUD-enabled devices must be configured to allow ACLs to be applied to each port. If the switch is stand-alone, then it will have to be configured to allow something like RADIUS or similar so that a controller device can send ACLs to the switch via an authorization transaction once the MUD profile has been processed.

For MUD to work properly, the switches MUST remove any dynamic configuration applied to a port when the connection on that port is dropped (such as when the cable to the port is disconnected). Once reconnected, a device will again issue a DHCP or similar request and the MUD behavior will begin again.

As an example, if a Layer-2 switch is used which can process DHCP requests by issuing RADIUS AAA requests to complete the port-level authorization, MUD process can occur by:

1. The switch adds the MUD URI to the RADIUS request (see [WEIS2017])

2. The RADIUS server passes the MUD URI to a MUD Controller

3. The returned MUD file is processed and the appropriate ACLs generated

4. The ACLs are encoded into the RADIUS Authorization response and returned to the switch

5. The switch receives the RADIUS Authorization, matches it to the port being provisioned, and applies the ACLs

3.4. Testing and Verification

In addition to the normal activities of validating through monitoring commands that ACLs have been applied as expected, the following items are suggested:

- o If one wants to understand what ACLs will be applied during a test of a particular device, one can read the MUD file to understand what access requirements it has and thus compare that with what ACLs get applied during the operation of the MUD protocol

- o The devices with MPDs attached to them should be checked to confirm the application of the expected ACLs and they are scoped to the appropriate ports

- o An ideal test would be to connect a MUD-enabled test client which will issue an appropriate network access negotiation via DHCP or whatever is appropriate for the NAS in use so that a full MUD File retrieval is triggered. The test client should then be used to try to both confirm connectivity to its explicity provisioned destination(s) while also verifying that it is not possible to reach sites outside the stipulated ACLs.

- o The MPD should be disconnected from the switch and the switch checked to verify that the ACLs are removed (which may not occur until another device is plugged into the same port)

3.5. Caching MUD Files

MUD Files may be cached by the MUD Controller. The MUD File itself indicates the minimum time between re-retrievals of a MUD File via the ‘‘cache-validity’’ attribute. When the MUD Controller is asked for a MUD File, if the URIs match a cached MUD File which is recent enough to be used, then that cached MUD File should be used. If not, then a valid MUD File MUST be retrieved by using the URI as a URL.

Note, however, that MUD files are very small. Additionally, MPDs will likely be installed into networks and then left running for long periods of time such that the number of MUD file requests will likely be small. Given those considerations, the value in caching MUD files, at least in the near term, is expected to be low.

4. Security Considerations

The bulk of this document describes the use of MUD to increase the security of a network. However, it is possible to compromise the effectiveness of MUD by attacking its behavior directly. This section discusses the known attacks and describes possible mitigations (all from the network operator’s perspective). This section also attempts to clarify the limits to which MUD is expected to perform in terms of increasing security.

The use of MUD is intended to increase the level of security in the network relative to its current state. If the network has no security protections in place, then MUD may improve the situation by limiting access to MUD-enabled devices, but the network may already be too permissively accessible to be secure. A common comment about MUD is that a compromised MUD File can allow a MUD-enabled device to access arbitrary parts of the network or to allow arbitrary access to the device. If the network had had no security to begin with, then the compromised MUD File will not have reduce the security in any meaningful way.

To put this another way, any network SHOULD be properly designed such that the minimum required access is granted to all parties involved. If this is done, then a bad MUD File can only result in too permissive access to and from a single device in the network.

Although MUD is still a new protocol, it is conceivable that an "ecosystem" around it will grow that will enable a level of security validation that is much more difficult without it. In particular, the published MUD Files could be analyzed by third parties to assess their contents and to make users aware of anomalies. Additionally, deviations in successive versions of MUD Files can be audited to detect surprising changes.

Another commonly-mentioned attack scenario is tampering with the MUD URI during device bring-up to cause a different MUD File to be fetched and applied in place of the correct, manufacturer-supplied file. The ramifications of such an attack are no different than that of a compromised MUD File. The mitigation against the attack is insure the use of secure means of receiving and processing the device’s advertisement of the MUD URI.

One other intriguing attack scenario is the spurious introduction of something akin to a "phantom" DHCP request with a MUD URI intended to coax the network infrastructure into fetching and acting on a MUD File, possibly without an actual device being present (or the "device" actually being a rogue software element running on a real device). In addition to mitigations already mentioned, port-level security should be used whenever possible with strict security policies to enable the detection of these rogue DHCP or other advertisements.

5. IANA Considerations

This document has no actions for IANA.

6. Normative References

[LEAR2017]
Lear, E., "Manufacturer Usage Description Specification", draft-ietf-opsawg-mud-03, January 05, 2017

[WEIS2017]
Weis, B., "RADIUS Extensions for Manufacturer Usage Description", draft-weis-radext-mud-00, October 25, 2016

7. Informative References

[RFC2882]
Mitton, D., "Network Access Servers Requirements: Extended RADIUS

Practices", RFC2882, July 2000

Authors' Addresses

Steven Rich
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134

Email: srich@cisco.com

Thorsten Dahm
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043

Email: thorstendlux@google.com

Network Working Group J. White
Internet-Draft D. Black
Intended status: Informational Dell EMC
Expires: September 9, 2017 J. Leung
Intel Corporation
March 9, 2017

YANG Data Center Baseline Switch Profile
draft-wbl-rtgwg-baseline-switch-model-01

Abstract

[Insert abstract here]

Status of This Memo

Copyright Notice

1. Introduction

Disclaimer - this is a -00 draft.

This is a normative profile for Baseline Switch Profile (send into IETF RTG) intended to be published as RFC on completion of DMTF spec to wrap Baseline Switch Profile.

2. What is a Redfish Baseline Switch?

The baseline switch profile contains basic system, interface, L2, and L3 configuration elements sufficient to set up the device for use in a controller based converged infrastructure environment.

The following list of IETF drafts, RFCs, and Redfish models will constitute the management interface to the baseline switch.

3. Core YANG RFCs

RFC6020 [1] provides the YANG modeling language definition.

RFC6991 [2] provides the Common YANG Data Types used by many other IETF YANG modules.

Interface management requires at set of RFCs to provide all relevant capabilities:

https://tools.ietf.org/html/rfc7223
https://tools.ietf.org/html/rfc7277
https://tools.ietf.org/html/rfc7224
https://tools.ietf.org/html/rfc7317

3.1. RFC7223 provides:

```
+--rw interfaces
|  +--rw interface* [name]
|     +--rw name                        string
|     +--rw description?                string
|     +--rw type                        identityref
|     +--rw enabled?                    boolean
|     +--rw link-up-down-trap-enable?   enumeration
+--ro interfaces-state
   +--ro interface* [name]
      +--ro name               string
      +--ro type               identityref
      +--ro admin-status       enumeration
      +--ro oper-status        enumeration
      +--ro last-change?       YANG:date-and-time
      +--ro if-index           int32
      +--ro phys-address?      YANG:phys-address
      +--ro higher-layer-if*   interface-state-ref
      +--ro lower-layer-if*    interface-state-ref
      +--ro speed?             YANG:gauge64
      +--ro statistics
         +--ro discontinuity-time    YANG:date-and-time
         +--ro in-octets?            YANG:counter64
         +--ro in-unicast-pkts?      YANG:counter64
         +--ro in-broadcast-pkts?    YANG:counter64
         +--ro in-multicast-pkts?    YANG:counter64
         +--ro in-discards?          YANG:counter32
         +--ro in-errors?            YANG:counter32
         +--ro in-unknown-protos?    YANG:counter32
         +--ro out-octets?           YANG:counter64
         +--ro out-unicast-pkts?     YANG:counter64
         +--ro out-broadcast-pkts?   YANG:counter64
         +--ro out-multicast-pkts?   YANG:counter64
         +--ro out-discards?         YANG:counter32
         +--ro out-errors?           YANG:counter32
```

3.2. RFC7277 adds:

```
+--rw if:interfaces
  +--rw if:interface* [name]
     ...
     +--rw ipv4!
     |  +--rw enabled?            boolean
     |  +--rw forwarding?         boolean
     |  +--rw mtu?                uint16
     |  +--rw address* [ip]
     |  |  +--rw ip               inet:ipv4-address-no-zone
     |  |  +--rw (subnet)
     |  |     +--:(prefix-length)
```

```
          |  |     |  +--rw ip:prefix-length?   uint8
          |  |     +--:(netmask)
          |  |        +--rw ip:netmask?         YANG:dotted-quad
          |  +--rw neighbor* [ip]
          |     +--rw ip                    inet:ipv4-address-no-zone
          |     +--rw link-layer-address    YANG:phys-address
          +--rw ipv6!
             +--rw enabled?            boolean
             +--rw forwarding?         boolean
             +--rw mtu?                uint32
             +--rw address* [ip]
             |  +--rw ip               inet:ipv6-address-no-zone
             |  +--rw prefix-length    uint8
             +--rw neighbor* [ip]
             |  +--rw ip                    inet:ipv6-address-no-zone
             |  +--rw link-layer-address    YANG:phys-address
             +--rw dup-addr-detect-transmits?   uint32
             +--rw autoconf
                +--rw create-global-addresses?        boolean
                +--rw create-temporary-addresses?     boolean
                +--rw temporary-valid-lifetime?       uint32
                +--rw temporary-preferred-lifetime?   uint32
```

AND

```
    +--ro if:interfaces-state
       +--ro if:interface* [name]
          ...
          +--ro ipv4!
          |  +--ro forwarding?   boolean
          |  +--ro mtu?          uint16
          |  +--ro address* [ip]
          |  |  +--ro ip               inet:ipv4-address-no-zone
          |  |  +--ro (subnet)?
          |  |  |  +--:(prefix-length)
          |  |  |  |  +--ro prefix-length?   uint8
          |  |  |  +--:(netmask)
          |  |  |     +--ro netmask?         YANG:dotted-quad
          |  |  +--ro origin?          ip-address-origin
          |  +--ro neighbor* [ip]
          |     +--ro ip                    inet:ipv4-address-no-zone
          |     +--ro link-layer-address?   YANG:phys-address
          |     +--ro origin?               neighbor-origin
          +--ro ipv6!
             +--ro forwarding?   boolean
             +--ro mtu?          uint32
             +--ro address* [ip]
             |  +--ro ip               inet:ipv6-address-no-zone
```

```
             |  +--ro prefix-length    uint8
             |  +--ro origin?          ip-address-origin
             |  +--ro status?          enumeration
             +--ro neighbor* [ip]
                +--ro ip                    inet:ipv6-address-no-zone
                +--ro link-layer-address?   YANG:phys-address
                +--ro origin?               neighbor-origin
                +--ro is-router?            empty
                +--ro state?                enumeration
```

3.3. RFC7224 provides:

The set of YANG identity statement for the IANA defined interface types.

3.4. RFC7317 provides:

- o System Identification
- o System Time Date
- o NTP
- o DNS Client

System Identification

```
     +--rw system
     |  +--rw contact?          string
     |  +--rw hostname?         inet:domain-name
     |  +--rw location?         string
     +--ro system-state
        +--ro platform
           +--ro os-name?       string
           +--ro os-release?    string
           +--ro os-version?    string
           +--ro machine?       string
```

System Time

```
     +--rw system
     |  +--rw clock
     |  |  +--rw (timezone)?
     |  |     +--:(timezone-name)
     |  |     |  +--rw timezone-name?     timezone-name
     |  |     +--:(timezone-utc-offset)
     |  |        +--rw timezone-utc-offset?   int16
     |  +--rw ntp!
     |     +--rw enabled?   boolean
     |     +--rw server* [name]
     |        +--rw name                string
     |        +--rw (transport)
     |        |  +--:(udp)
     |        |     +--rw udp
     |        |        +--rw address    inet:host
     |        |        +--rw port?      inet:port-number
     |        +--rw association-type?   enumeration
     |        +--rw iburst?             boolean
     |        +--rw prefer?             boolean
     +--ro system-state
        +--ro clock
           +--ro current-datetime?      YANG:date-and-time
           +--ro boot-datetime?         YANG:date-and-time
```

DNS Client

```
     +--rw system
        +--rw dns-resolver
           +--rw search*    inet:domain-name
           +--rw server* [name]
           |  +--rw name    string
           |  +--rw (transport)
           |     +--:(udp-and-tcp)
           |        +--udp-and-tcp
           |           +--rw address    inet:ip-address
           |           +--rw port?      inet:port-number
           +--rw options
              +--rw timeout?    uint8
              +--rw attempts?   uint8
```

User Authentication

```
     +--rw system
        +--rw authentication
           +--rw user-authentication-order*   identityref
           +--rw user* [name]
              +--rw name        string
              +--rw password?   ianach:crypt-hash
              +--rw authorized-key* [name]
                 +--rw name         string
                 +--rw algorithm    string
                 +--rw key-data     binary
```

4. Additional YANG models

In addition to the above RFCs, the baseline switch models needs to cover:

- o VLANs
- o ACLs
- o Syslog

The following lists of IETF drafts sets our recommendation to cover the above three areas.

4.1. VLAN and interface extensions:

To handle VLANs and with related interface configuration the following YANG models are under evaluation.

- o https://tools.ietf.org/html/draft-ietf-netmod-intf-ext-yang-03
- o https://tools.ietf.org/html/draft-wilton-intf-vlan-yang-00.txt ## ACL To handle ACL configuration the following YANG model is under consideration.
- o https://tools.ietf.org/html/draft-ietf-netmod-acl-model-09

4.2. Syslog

To handle configuration and access to syslog the following YANG model is under consideration.

- o https://tools.ietf.org/html/draft-ietf-netmod-syslog-model-11

5. Applicable Redfish system management models

The following standard Redfish systems management models apply to the baseline network switch profile. Reference: Redfish schema index [3]. The use of these Redfish management models allows a converged infrastructure manager to have a consistent view of server, storage and network systems.

- o Chassis
- o ComputerSystem
- o Manager
- o ManagerAccount
- o Power
- o Thermal
- o SoftwareInventory plus UpdateService
- o Event configuration using Event, EventDestination, and Event Service
- o Access to logs using LogEntry, and LogService
- o Management interface configuration using EthernetInterface and related
- o Console configuration using SerialInterface
- o PrivilegeRegistery and Privileges

Where YANG and Redfish overlap, the commonality of YANG vs Redfish is TBD.

6. Overall Baseline Switch Profile Structure

```
./redfish/v1/Systems
./redfish/v1/Chassis
./redfish/v1/NetworkDevices/BaselineSwitch/...
... other redfish resource blocks...
(resource from RFCs and Redfish bullet list, above)
```

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2. URIs

[1] https://tools.ietf.org/html/rfc6020

[2] https://tools.ietf.org/html/rfc6991

[3] http://redfish.dmtf.org/redfish/schema_index

Authors' Addresses

Joseph White
Dell EMC

Email: joseph.l.white@dell.com

David Black
Dell EMC

Email: david.black@dell.com

John Leung
Intel Corporation

Email: john.leung@intel.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 9, 2017

J. White
D. Black
Dell EMC
J. Leung
Intel Corporation

March 9, 2017

YANG Baseline Switch Profile Background
draft-wbl-rtgwg-yang-ci-profile-bkgd-01

Abstract

A YANG device profile is primarily a group of YANG models that are appropriate for use with a particular class of device or in specific device roles. This document provides background and describes the rationale for a baseline data center switch device profile, e.g., for top-of-rack switches in data center converged infrastructure. This rationale is based on the reuse of YANG models by the DMTF Redfish standard for management of converged infrastructure, but the resulting YANG device profile is intended to be useable by any YANG-based network management approach or protocol, as opposed to being specific to Redfish.

1. Introduction

Disclaimer - this is a -00 draft, whose primary goal is to capture the rationale for use of YANG for Redfish network management and the desirability of a baseline data center network switch profile, including providing technical background on the Redfish standard and its approach to network management. The draft content is not polished, and the organization of the material is likely to change.

A YANG device profile is primarily a group of YANG models that are appropriate for use with a particular class of device or in specific device roles. This document provides background and describes the rationale for a baseline data center switch device profile, e.g., for top-of-rack switches in data center converged infrastructure. The rationale is based on the reuse of YANG models by the DMTF Redfish standard for management of converged infrastructure, but the resulting YANG device profile is intended to be useable by any YANG-based network management approach or protocol; it is not intended to be Redfish-specific.

In support of this rationale, this document provides background on how YANG is used in the Redfish management framework; the YANG modules are translated to Redfish schema definitions in order to enable reuse of the models are with Redfish-defined management protocols and related functionality.

2. Motivation

A common management framework with accompanying set of protocols and device models is desirable in systems management use cases. A good example of this is in a converged infrastructure deployment within a data center. Applications, servers, storage, appliances, and networking elements are assembled to create a combined coherent platform. For the networking components in such an environment, there are platform management elements that are common with other types of systems such as thermal monitoring, physical enclosure, fans, and power supplies, as well as networking specific management elements to control the forwarding and filtering of network packets or the networking services. The common elements should be accessed and managed in a single way across all systems within the deployment.

Management, orchestration, and control of such a system benefits from a single approach that enables unified tools sets and simplifies operations.

The networking specific configuration within the converged infrastructure environment only needs a subset of all the possible networking protocols and services giving the converged controller only the minimum spanning control surface in terms of the models it can access. A breakdown of the needs of such a switch result in about 5-10 YANG modules (see Appendix A). These 5-10 modules should lead to common YANG-based data center network switch management across vendors and products.

As a contrast, a full function edge router would need many more protocols and services along with full function virtualization resulting in the use of about 80 YANG modules.

2.1. Redfish Background

The DMTF (Distributed Management Task Force) Redfish standard is becoming the common standard for converged infrastructure (CI) management. Converged Infrastructure consists of rack-scale (partial or full-rack) integrated compute, network and storage infrastructure that is procured and deployed as rack scale systems.

Redfish data center storage management functionality has been extended in partnership with SNIA (Storage Networking Industry Association) resulting in the recently released Swordfish specification that extends Redfish for networked storage and storage network management. The authors hope to work with the IETF to extend and align Redfish network management with IETF’s YANG framework.

Redfish is a management standard using a data model representation inside of a RESTful interface. The model is expressed in terms of a standard, machine-readable schema, with the payload of the messages being expressed in JSON.

Because it is a model-based API, Redfish is capable of representing a variety of implementations via a consistent interface. It has mechanisms for managing data center resources, handling events, long lived tasks and discovery, as well.

In Redfish, every URL represents a resource. This could be a service, a collection, an entity or some other construct. But in RESTful terms, these URIs point to resources and Redfish clients interact with these resources. For example, the content of a resource, in JSON format, is returned when the Redfish client performs a HTTP GET.

OData is an OASIS standard for expressing the schema of a JSON document. OData allows a fuller description of the JSON document, than json-schema. The format of OData schema is specified in CSDL (Common Schema Definition Language).

OData also describes directives that can appear on the URI to control the contents of the HTTP response. In Redfish, these directives (i.e. $top and $skip) are stated as 'should'. Networking extension may change the requirement to 'shall'.

Redfish releases include both OData and json-schema schema. With json-schema, users can take advantage of its larger tool chain.

Additional information about OData can be found at the OData site [1].

Additional information about Redfish can be found at DMTF's Redfish site [2]. Specifically, the Redfish White Paper [3] provides a good overview.

2.2. YANG and Redfish

Within the networking world, YANG has become the preferred management framework. YANG expresses each section of the overall model as a module containing a tree of nodes where each node is either a container node that builds the hierarchy or a leaf node containing data elements for the model. Redfish network management leverages YANG as the core model definition language to maintain consistency with other YANG based network management approaches. Using a common model structure results in equivalent data elements yielding the same data or content when accessed via different interfaces or protocols.

Redfish's network management supports this consistency by reusing YANG modules as Redfish models for network functionality and services, mechanically translating those modules to the Redfish interface, based on HTTP, JSON, and OData.

The Redfish approach to network management enables definitions of a specific system views or profiles that are aligned with the configuration functionality required in a specific scenario or for a specific class of network devices. A particularly relevant example is a baseline switch model description with a minimum set of model elements; this is useful when configuring a switch within a larger converged infrastructure system. The model elements of the baseline switch should be the smallest set of models necessary to configure a data center switch in a converged infrastructure environment; the corresponding set of YANG modules would be a simple data center network device profile. A more complex network router might need tunnel models, overlay models, extensive QoS models, and other elements.

The top level resource structure of Redfish is show below.

./redfish/v1/Systems
./redfish/v1/Chassis
./redfish/v1/NetworkDevices
(other redfish resources)

Within this structure a network switch is viewed as a data center element for its common subsystems such as chassis, power, thermal, cooling, management access setup, etc while the network modeling is specified within the instances of the NetworkDevices[] collection. Each member of the NetworkDevices[] collection has implements its own set translated YANG modules. For consistency, the set of modules grouped under a NetworkDevice instance can follow one of multiple standard groupings expressed as a profile to manage different classes of equipment and satisfy different use cases. Two profile examples are the basic switch and virtualized edge router.

2.3. YANG mapping to Redfish

The notion of schema is different in Redfish and YANG.

In YANG, a schema is device specific. The user determines the YANG modules utilized by a system, and may consult a module library as part of doing so (e.g., RFC7895 [4]). The YANG schema is realized as a set of YANG modules, each with a prescribed node tree structure.

In Redfish, there is one schema that encompasses the entire namespace. In other words, Redfish has a global namespace for its schema, of which the device implements a subset. The Redfish schema is realized as resources accessed via a URI, so the namespace contains the information about which YANG modules are utilized. The OData directives $expand and $filter allows the client to discovery this directly from the parent namespace node above the modules.

That functionality obviates any Redfish need to use YANG module combination techniques such as YANG Schema-mount [5].

Despite these differences, the proposed profiles should be usable by both YANG based protocols (e.g., NETCONF, RESTCONF) and Redfish, as the core content of each profile is a set of YANG modules.

To allow Redfish to manage network devices, the YANG modules needs to be translated into OData CSDL (Common Schema Definition Language). The translation is specified in the YANG-to-Redfish Mapping

Specification [6]. The translation has the following characteristics:

- o Includes, imports, and augments, are compiled out to create a single consistent schema block constituting a particular instance of a NetworkDevice.

- o The YANG node tree layout is reflected in the URI layout

- o The individual YANG container nodes and list nodes are rendered as resources with the YANG tree hierarchy reflected as navigation properties.

Access to the YANG data model elements uses a Redfish JSON accessed via a provider on the URI target.

Leaf nodes representing common back end system "features or elements" return consistent data independent of node name and network device hierarchy.

The NetworkDevices[] collection allows

- o Multiple co-existing and consistent views onto a system.

  * Horizontally extensible

  * Vertical hierarchy allowing for control interface delegation

- o This is similar to a "view class" or facade approach in software.

2.4. Example Mapping

The following shows the resource which results from mapping RFC7223 (ietf_interface module) to the Redfish schema. Below is a fragment of the data model from the RFC.

```
+--rw interfaces
| +--rw interface* [name]
| +--rw name string
| +--rw description? string
| +--rw type identityref
| +--rw enabled? boolean
| +--rw link-up-down-trap-enable? enumeration
+--ro interfaces-state
+--ro interface* [name]
+--ro name string
+--ro type identityref
+--ro admin-status enumeration
```

The translation to Redfish CSDL is performed using the RFC's YANG code. The translation will generate the CSDL files for the ietf_interfaces resource and each YANG container. The path to these resources mirror the above data model.

```
./redfish/v1/NetworkDevices/Switch1
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces/interfaces
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces/interfaces/ethernet1
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces/interfaces_state
...
```

A HTTP GET of the "ethernet1" singleton resource will return the following JSON document. Note that each property from the above data model is present in the resource.


```
{
    "Id": "ethernet1",
    "Name": "ethernet1",
    "Description": "Ethernet interface on slot 1",
    "type": "iana_if_type:ethernetCsmacd",
    "enabled": "true",
    "link_up_down_trap_enable": "true"

    "@odata.context":
        "/redfish/v1/$metadata#ietf_interfaces.interfaces.interface.
            interface",
    "@odata.type": "#interface.v1_0_0.interfaces",
    "@odata.id":
        "/redfish/v1/NetworkDevices/Switch1/ietf_interfaces
            /interfaces/ethernet1"
}
```


The three properties at the end of the JSON document are OData annotations.

3. Security Considerations

Redfish also improves security control since there is a single point of management contact for a device to control all of its functions.

(Additional security discussion will be provided later.)

4. Appendix A

YANG models needed to managed a network switch:

o RFC7223 (Interfaces)

- o RFC7224 (IANA)
- o RFC7277 (IPv4, IPv6)
- o RFC7317 (System Identification, Time-Date, NTP)
- o VLANs
- o ACLs
- o Syslog

5. Appendix B

The following describes how the Redfish NetworkDevices[] collection resource allows multiple co-existing and consistent views onto a system.

As an example, a router could have the following:

```
//redfish.example.net/redfish/v1/NetworkDevices/masterRouter
//redfish.example.net/redfish/v1/NetworkDevices/vrf1
//redfish.example.net/redfish/v1/NetworkDevices/vrf2
```

In this example, masterRouter represents the complete system with all interfaces, all tables, all system level configuration, and a model structure for assigning resources to virtual instances. The resources, vrf1 and vrf2, represent a particular partitioning of the system to create virtual router instances each assigned a subset of the total resource pool.

The above structure has similarities with that expressed by the device model from the following references:

- o https://tools.ietf.org/html/draft-ietf-rtgwg-device-model-01
- o https://tools.ietf.org/html/draft-ietf-rtgwg-ni-model-01
- o https://tools.ietf.org/html/draft-ietf-rtgwg-lne-model-01
- o https://tools.ietf.org/html/draft-ietf-netmod-schema-mount-03

In these references a Network Device contains Logical Network Elements which, in turn, contain Network Instances. From the device model reference, the Network Device represents the system as a whole. The Logical Network Element represents a partition of a physical system. The Logical Network Element represents a VRF or VSI (virtual switching instance).

The Redfish NetworkDevices collection resource would map this modeling approach by using an element of the collection for the Network Device and one for each of the Logical Network Elements and Network Instances. These collection elements would add references at the NetworkDevices element level to map the containment of of the device model. The overall ./redfish/v1/ root maps to the Routing Area Network Device.

6. Appendix C

The following is the ietf_interfaces.interfaces.interface_v1.xml CSDL metadata file, which is referenced in @odata.context annotation in the example mapping. The entity type referenced in the @odata.type annotation is in the second Namespace.

When mapping YANG code to CDSL, values are mapped to existing OData core properties, when possible. Otherwise, new annotations are defined in RedfishYangExtensions.xml. This file is referenced at the beginning of the document.


```
<?xml version="1.0" encoding="UTF-8"?>
<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx"
        Version="4.0">
    <Edmx:Reference
        Uri="http://docs.oasis-open.org/odata/odata/v4.0/cs01/
            vocabularies/Org.OData.Core.V1.xml">
        <Edmx:Include Alias="Odata" Namespace="Org.OData.Core.V1"/>
    </Edmx:Reference>
    <Edmx:Reference
        Uri="http://docs.oasis-open.org/odata/odata/v4.0/cs01/
            vocabularies/Org.OData.Capabilities.V1.xml">
        <Edmx:Include Alias="Odata"
            Namespace="Org.OData.Capabilities.V1"/>
    </Edmx:Reference>
    <Edmx:Reference
        Uri="http://redfish.dmtf.org/schemas/v1/
            RedfishYangExtensions.xml">
        <Edmx:Include Alias="Redfish.Yang"
            Namespace="Redfish.Yang"/>
    </Edmx:Reference>

    <Edmx:DataServices>

    <Schema Namespace="interface"
            xmlns="http://docs.oasis-open.org/odata/ns/edm" >
        <EntityType Name="interface"
                BaseType="Resource.v1_0_0.Resource">
            <Annotation Term="OData.Description"
```

```
                    String="<manual input>." />
                <Annotation Term="OData.AdditionalProperties"
                    Bool="False"/>
            </EntityType>
        </Schema>

        <Schema Namespace="interface.v1_0_0"
                xmlns="http://docs.oasis-open.org/odata/ns/edm" >
            <EntityType Name="interface" BaseType=
                    "ietf_interfaces.interfaces.interface.interface" >
                <Annotation Term="OData.Description"
                    String="<manual input>." />
                <Annotation Term="OData.AdditionalProperties"
                    Bool="False"/>
                <Annotation Term="Redfish.Yang.NodeType"
                    EnumMember="Redfish.Yang.NodeTypes/list" />
                <Annotation Term="Redfish.Yang.key"
                    String=" the yang key string"/>
                <Key>
                    <PropertyRef Name="name"/>
                </Key>
                <Property Name="name" Type="Edm:String">
                    <Annotation Term="OData.Description"
                        String="..." />
                    <Annotation Term="OData.Permissions"
                        EnumMember="OData.Permissions/Read"/>
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf"/>
                    <Annotation Term="Redfish.Yang.YangType"
                        String="string" />
                </Property>
                <Property Name="description" Type="Edm:String">
                    <Annotation Term="OData.Description"
                        String="..." />
                    <Annotation Term="OData.Permissions"
                        EnumMember="OData.Permissions/Read"/>
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf" />
                    <Annotation Term="Redfish.Yang.YangType"
                        String="string" />
                    <Annotation Term="Redfish.Yang.reference"
                        String="RFC 2863: The Interfaces Group..." />
                </Property>
                <Property Name="type" Type="Edm:String">
                    <Annotation Term="OData.Description" String="..."/>
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf" />
                    <Annotation Term="Redfish.Yang.YangType"
```

```
                        String="identityref"/>
                    <Annotation Term="Redfish.Yang.base"
                        String="interface-type"/>
                    <Annotation Term="Redfish.Yang.mandatory"
                        EnumMember="Redfish.Yang.Mandatory/true"/>
                    <Annotation Term="Redfish.Yang.reference"
                        String="RFC 2863: The Interfaces Group..." />
                </Property>
                <Property DefaultValue="true" Name="enabled"
                        Type="Edm:Boolean">
                    <Annotation Term="OData.Description"
                        String="This leaf contains..." />
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf" />
                    <Annotation Term="Redfish.Yang.YangType"
                        String="boolean"/>
                    <Annotation Term="Redfish.Yang.reference"
                        String="RFC 2863: The Interfaces..."/>
                </Property>
                <Property Name="link_up_down_trap_enable"
                        Type="Edm:Enumeration">
                    <Annotation Term="OData.Description"
                        String="Controls whether..."/>
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf"/>
                    <Annotation Term="Redfish.Yang.YangType"
                        String="enumeration"/>
                    <Annotation Term="Redfish.Yang.if_feature"
                        String="if-mib"/>
                    <Annotation Term="Redfish.Yang.reference"
                        String="RFC 2863: The Interfaces..." />
                    <EnumType
                            Name="link_up_down_trap_enableEnumeration">
                        <Member Name="enabled" Value="1">
                            <Annotation Term="Redfish.Yang.enum"
                                String="enabled"/>
                        </Member>
                        <Member Name="disabled" Value="2">
                            <Annotation Term="Redfish.Yang.enum"
                                String="disabled"/>
                        </Member>
                    </EnumType>
                </Property>
            </EntityType>
        </Schema>

        </Edmx:DataServices>
```

```
</edmx:Edmx>
```

7. Appendix D

The following is the IETF YANG source XML from RFC7223 used for the example mapping.

```
<CODE BEGINS> file "ietf-interfaces@2014-05-08.yang"
module ietf-interfaces {
    namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";
    prefix if;
    import ietf-yang-types {
        prefix yang;
    }
    organization
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
    . . .
```

After the typedef, identity, and feature statements, the data model is defined. Below is the fragment that becomes ietf_interfaces.interfaces.interface_v1.xml.

```
    /*
     * Configuration data nodes
     */
    container interfaces {
        description
            "Interface configuration parameters.";
        list interface {
            key "name";
            description
                "The list of configured interfaces...";
        leaf name {
            type string;
            description
                "The name of the interface...";
        }
        leaf description {
            type string;
            description
                "A textual description of the interface...";
            reference
                "RFC 2863: The Interfaces Group MIB - ifAlias";
        }
        leaf type {
            type identityref {
                base interface-type;
            }
```

```
                mandatory true;
                description
                    "The type of the interface...";
                reference
                    "RFC 2863: The Interfaces Group MIB - ifType";
            }
            leaf enabled {
                type boolean;
                default "true";
                description
                    "This leaf contains the configured,...";
                reference
                    "RFC 2863: The Interfaces Group MIB - ifAdminStatus";

            leaf link-up-down-trap-enable {
                if-feature if-mib;
                type enumeration {
                    enum enabled {
                        value 1;
                    }
                    enum disabled {
                        value 2;
                    }
                }
                description
                    "Controls whether linkUp/linkDown SNMP...";
                reference
                    "RFC 2863: The Interfaces Group MIB -
                        ifLinkUpDownTrapEnable";
            }
        }
        . . .
```

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. URIs

[1] http://odata.org

[2] http://dmtf.org/redfish

[3] http://www.dmtf.org/sites/default/files/standards/documents/DSP2044_1.0.0.pdf

[4] http://www.rfc-editor.org/info/rfc7895

[5] https://tools.ietf.org/html/draft-ietf-netmod-schema-mount-03

[6] http://www.dmtf.org/sites/default/files/standards/documents/DSP0271_0.5.6.pdf

Authors' Addresses

Joseph White
Dell EMC

Email: joseph.l.white@dell.com

David Black
Dell EMC

Email: david.black@dell.com

John Leung
Intel Corporation

Email: john.leung@intel.com

Network Working Group Y. Zheng
Internet-Draft China Unicom
Intended status: Informational S. Xu
Expires: September 14, 2017 D. Dhody
Huawei Technologies
March 13, 2017

Usecases for Network Artificial Intelligence (NAI)
draft-zheng-opsawg-network-ai-usecases-00

Abstract

This document discusses the scope of Network Artificial Intelligence (NAI), and the possible use cases that are able to demonstrate the advantage of applying NAI.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Current networks have become much more dynamic and complex, and pose new challenges for network management and optimization. For example, network management/optimization should be automated to avoid human intervention (and thus to minimize the operational expense). Artificial Intelligence (AI) and Machine Learning (ML) is a promising approach to realize such automation, and can even do better than human beings. Furthermore, the population of Software-Defined Networks (SDN) paradigm makes the application of Artificial Intelligence in networks possible, since the SDN controller has the complete knowledge of the network status and can control behavior of network nodes to implement AI decisions.

AI and ML technologies can learn from historical data, and make predictions or decisions, rather than following strictly static program instructions. They can dynamically adapt to a changing situation and enhance their own intelligence with by learning from new data. It can learn and complete complicated tasks. It also has potential in the network technology area especially with SDN and Network Function Virtualization (NFV).

This document presents the concept of Network Artificial Intelligence. It first discusses the scope of Network Artificial Intelligence (NAI). And then Some use cases are discussed to demonstrate the advantage of applying NAI.

2. NAI Architecture

The definition of the architecture of NAI could be refer to [I-D.li-rtgwg-network-ai-arch]. In the architecture of NAI, central controller is the core part of Network Artificial Intelligence which can be called as 'Network Brain'. The Network Telemetry and Analytics (NTA) engines can be introduced acompanying with the central controller. The Network Telemetry and Analytics (NTA) engine inclues data collector, analytics framework, data persistence, and NAI applications.

```
                ^                                   ^
             (4)|                                   |(4)
+---------------|--------------+    +---------------|--------------+
| Domain 1      |              |    |               |     Domain 2 |
|        +------------+        |    |        +------------+        |
|        |  Central   |        |    |        |  Central   |        |
|     (1)| Controller |----------------------| Controller |(1)     |
|        |    with    |        |    |        |    with    |        |
|        |     NTA    |        |    |        |     NTA    |        |
|        +------------+        |    |        +------------+        |
|          /         \         |    |          /         \         |
|     (3)/            \        |    |         /           \(3)     |
|       /              \       |    |        /              \      |
| +--------+        +--------+ |    | +--------+        +--------+ |
| |        |        |        | |    | |        |        |        | |
| |Network | ...... |Network | |    | |Network | ...... |Network | |
| | Device |  (2)   | Device | |    | | Device |  (2)   | Device | |
| |    1   |        |    N   | |    | |    1   |        |    N   | |
| +--------+        +--------+ |    | +--------+        +--------+ |
|                              |    |                              |
+------------------------------+    +------------------------------+
```

Figure 1: An Architecture of Network Artificial Intelligence(NAI)

3. NAI Use Cases

3.1. Traffic Predication and Re-Optimization/Adjustment

This subsection introduces the Path Computation Element (PCE) [RFC4655] use cases in wide area networks (WAN). In PCE scenario, network data collection is realized through the control plane protocols such as PCE protocol (PCEP) and BGP-LS [RFC7752] protocol and data are passed to the PCE application. PCEP receives the state of Label Switched Path (LSP) from the network, and BGP-LS receives the topology information from the network. If network telemetry is used, traffic information can be received from the network as well directly at the NTA engine using protocols such as gRPC.

PCE application (APP) only maintains the latest information. To enable NAI, history of all LSP and topology changes is stored in external data repository. Further traffic monitoring data could also be collected and stored, if network telemetry is used. There are two usecases in the application scenarios: (1) reroute/re-optimize using the historical trend and predications from AI; (2) traffic congestion avoidance and AI-enabled auto-bandwidth adjustment.

For the usecase (1), the analytics component in NTA (Network Telemetry and Analytics), can use stored data to build models to predict impact of network events and state of the LSPs. For example, it can use historical trends to guide path computation to include/exclude specific links. Finding correlations between data, finding anomalies and data visualization are also possible.

The analytics component in NTA can also use stored data to detect and predict network events and request PCE to take necessary actions. For example, it can use network bandwidth utilization historical trends to request for re-optimizations.

For the usecase (2), with network telemetry, the NTA can collect per-link and per-LSP traffic flow using gRPC from network. Such network telemetry data includes statistics for tunnels, links, bandwidth reservations, actual usage, delay, jitter, packet loss, etc. Meanwhile, it also collects data regarding network events and its impact on traffic flows. The analytics component can use telemetry data to build traffic models to predict traffic congestion when new years or sporting events are coming. According to the congestion prediction, the PCE app could reroute traffic to avoid congested links. Besides the case, NTA can also perform predication and make necessary changes to network. In particular, the PCE APP performs bandwidth usage prediction (i.e., bandwidth calendaring) by looking at the historical trends of all sampled data instead of the instant sampled data. The collected data are traffic engineering data base (TEDB) and LSP-DB, and can also include scheduling information. In addition, the collected data also include auto-bandwidth related changes under particular network events. Using machine learning algorithm, the analytics component is able to correct such changes with the events, and predicts network events and their impact.

3.2. Route Monitoring and Analytics

This subsection introduces the BGP Monitoring Protocol (BMP) [RFC7854] use case in wide area networks (WAN). The BGP protocol is known for its flexibility and ability to manage a large number of neighbors and routes. It is also the basis for many overlay services such as L3VPN, L2VPN and so on. The BMP protocol can be used by the controller to monitor BGP protocol neighbor status and routing information on the routers.

According to [RFC7854], BMP client located in the router collects BGP neighbor status, routes for each neighbor, and events defined by the user. And then it passes the informations through the BMP protocol to the management station located on the controller. Based on BMP monitoring of BGP, there are three use cases: (1) BGP Route Leaks Monitoring; (2) BGP Hijacks Monitoring; (3) Traffic Analytics.

Route leaks involve the illegitimate advertisement of prefixes, blocks of IP addresses, which propagate across networks and lead to incorrect or suboptimal routing. For case (1), based on BMP, NAI apps can analyze BGP route leaks.

For case (2), by manipulating BGP, data can be rerouted in an attacker’s favor out them to intercept or modify traffic.If the malicious announcement is more specific than the legitimate one, or claims to offer a shorter path, the traffic may be directed to the attacker.By broadcasting false announcements, the compromised router may poison the RIB of its peers.After poisoning one peer, the malicious routing information could propagate to other peers, to other Autonomous Systems, and onto the interactive Internet. Based on monitoring BGP routes, ML algorithms can be trained to determine when a hijack has taken place and take necessary actions.

In case (3), with BMP protocol providing BGP changes, together with Telemetry providing network traffic information, The NAI Apps can analyze traffic trends, predict traffic changes, and do traffic optimizing.

3.3. Multilayer Fault Detection In NFV Framework

The high reliability and high availability required for carrier-class applications is a big challenge in virtualized and software-based environment where failures are normal in a software-based environment. The interdependence between NFV’s abstraction levels and virtual resources is complex as shown in Fig.. The dynamic characteristics of the resources in the cloud environment make it difficult to locate the fault. So multilayer fault detection for NFV networks and cloud environment will be very useful.

```
                         +--------------------+
                         |      Central       |
                         |     Controller     |
                         |        with        |
                         |         NTA        |
                         +--------------------+
                             |      |      |
                             |      |      |
                             |      |      |
                             V      V      V
        +-------------------------------------------------------+
        |                                                       |
        |      +-----------+  +-----------+  +-----------+      |
        |      |    VNF1   |  |    VNF2   |  |    VNF3   |      |
        |      +-----|-----+  +-----|-----+  +-----|-----+      |
        |            |              | VN-NF        |            |
        |    +-------|--------------|--------------|-------+    |
        |    | NFVI                                        |    |
        |    | +-----------+  +-----------+  +-----------+ |    |
        |    | |  Virtual  |  |  Virtual  |  |  Virtual  | |    |
        |    | | Computing |  |  Storage  |  |  Network  | |    |
        |    | +-----------+  +-----------+  +-----------+ |    |
        |    | +-----------------------------------------+ |    |
        |    | |        VIRTUALIZATION LAYER             | |    |
        |    | +--------------------|--------------------+ |    |
        |    |                VI-Ha |                      |    |
        |    |+---------------------|---------------------+|    |
        |    ||                         Hardware Resouces ||    |
        |    ||+-----------+  +-----------+  +-----------+||    |
        |    ||| Computing |  |  Storage  |  |  Network  |||    |
        |    |||  Hardware |  |  Hardware |  |  Hardware |||    |
        |    ||+-----------+  +-----------+  +-----------+||    |
        |    |+-------------------------------------------+|    |
        |    +---------------------------------------------+    |
        |                                                       |
        +-------------------------------------------------------+
```

Figure 2 NAI in Multi-layer NFV Framework

For the virtualization layer, CPU performance, memory usage, interface bandwidth and other KPI indicators can be monitored. At the same time resource occupancy and the life cycle of NVF software process can also be monitored. Through the NAI, the relevant statistical data in multiple levels can be analyzed and the models can be setup to locate the root cause for the possible fault in the multi-layer environment.

3.4. Data Center Network Use Cases

Traditionally, data center networks have comprised a large number of switches and routers that direct traffic based on the limited view of each device. With help of SDN/NFV the data center networks are more agile and dynamic to changing usage and traffic patterns. The real-time traffic data and usage can be used to make the data center management and operations intelligent.

Various protocols such as sFLOW, IPFIX could be used to get the port statistics as well as traffic sampling. Over time this information can help build the traffic usage models on a per port and per flow basis. With historical data as the base the NTA engine can predict the traffic usage and make necessary instructions to the SDN controller or NFV orchestrator. These instructions could be reroute a flow to avoid a congested port or scale-in another switch to share load based on the predicted traffic demand.

The NTA engine should find correlation between the various network data to build models and predict the impact of network events, congestions, network utilization patters etc. Further NTA could detect anomalies based on the historical patterns and help in root cause analysis. The policy framework can be enhanced to consider the analytics.

NTA engine could also get the usage and health information from the Host (servers). Correlation between this information with the information received from network could help in finding security flows and anomalies when the information does not match.

3.4.1. Service Function Chaining

This sub section introduces how to apply NAI to SFC scenario to intelligently reroute/re-optimize the service chains; increase utilization for both Service Functions(SF) and network; intelligent selection of the Service Function Path (SFP) based on data traffic trends.

As per [RFC7665], Service function chaining (SFC) enables the creation of composite (network), services that consist of an ordered set of SFs that must be applied for specific treatment of received packets and/or frames and/or flows selected as a result of classification The SFs of chain are connected using a service function forwarder (SFF), which is responsible for forwarding traffic to one or more connected SFs according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF.

The various network telemetry information like delay, jitter, packet loss from the network and the CPU/memory usage utilizations from the SFs, can be collected using sFLOW/gRPC protocol and stored in persistent data repository. The analytics component in NTA can use stored data to build statistics models to predict the impact on various Service Function Paths due to network events, traffic and state of the SFPs and instruct the SDN controller to take necessary actions SDN controller can calculate new paths/reroute the SFC path to avoid congested Ports/SFFs or overloaded SFs. This correlation of application analytics from the SFs and the network analytics from the SFFs could enhance the intelligent management of the service chains for the operators.

The usage and traffic pattern over time can help increase the utilization of SF as well as the underlay network.

4. Contributors

The following people have substantially contributed to the usecases of NAI:

Lizhao You
Huawei
Email: youlizhao@huawei.com

Kalyankumar Asangi
Huawei
Email: kalyana@huawei.com

5. Security Considerations

TBD

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgement

Thanks to Li Zhenbin and Liu Shucheng for their comments and contribution.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[I-D.li-rtgwg-network-ai-arch] Li, Z. and J. Zhang, "An Architecture of Network Artificial Intelligence(NAI)", draft-li-rtgwg-network-ai-arch-00 (work in progress), October 2016.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <http://www.rfc-editor.org/info/rfc4655>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <http://www.rfc-editor.org/info/rfc7752>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <http://www.rfc-editor.org/info/rfc7854>.

Authors' Addresses

Yi Zheng
China Unicom
No.9, Shouti Nanlu, Haidian District
Beijing 100048
China

Email: zhengyi39@chinaunicom.cn

Xu Shiping
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
P.R. China

Email: xushiping7@huawei.com

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

Email: dhruv.ietf@gmail.com